

Corrigés Servlets JSP

SOMMAIRE

1. PREAMBULE.....	4
2. MISE EN ŒUVRE DE SERVLETS.....	5
2.1. AFFICHAGE D'UN TEXTE	5
2.2. AFFICHAGE D'UN CONTENU DE BOUCLE	7
2.3. SAISIE D'UN FORMULAIRE.....	8
2.4. MANIPULATION DU WEB.XML.....	12
2.4.1. Copier/Coller du Servlet.....	12
2.4.2. Exécution du Servlet 2	13
2.4.3. Ajout parametre d'application fichier web.xml	13
2.4.4. Parametre du servlet	15
2.4.5. Informations sur le serveur.....	16
3. COMMUNICATION ENTRE SERVLETS.....	17
3.1. GESTION DE LA NAVIGATION	17
3.1.1. Le contrôleur	17
3.1.2. Le Formulaire de login.....	19
3.1.3. Le Formulaire de réservation	20
3.2. GESTION DES SESSIONS	21
3.3. LES JAVABEANS NECESSAIRES A L'APPLICATION	21
3.3.1. Les beans	21
3.3.2. La gestion des beans dans les servlets.....	23
4. BIBLIOTHEQUE EN LIGNE.....	25
4.1. REMPLACER LES SERVLETS PAR DES JSP	25
4.2. FONCTIONNEMENT DU CONTROLEUR.....	27
4.3. MODULARITE DES PAGES	28
4.3.1. Exemple de code	29
4.3.2. Entête et pied de page.....	29
4.3.2.1. Entête	29
4.3.2.2. Pied de page	29
4.3.3. Servlet Contrôleur	29
4.3.4. FormulaireS.....	30
4.3.5. Formulaire de reservation de livres.....	31
4.3.6. Page de fin de session.....	31
4.3.7. Page d'erreur.....	32
4.3.8. Classe Livre.....	33

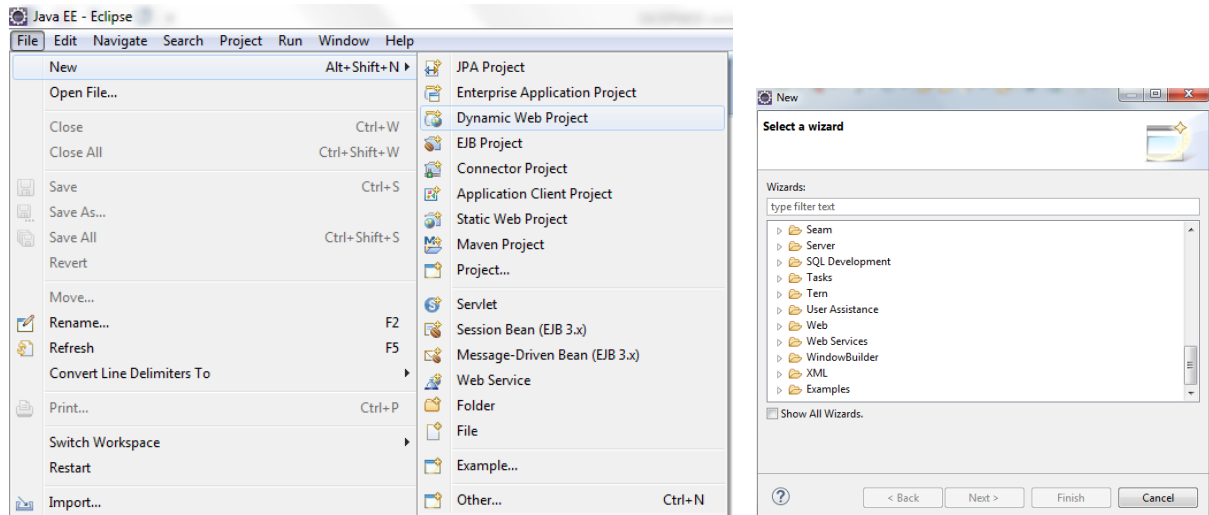
4.3.9.	Classe Client.....	34
5.	REALISATION DE BALISES PERSONNALISEES	35
5.1.	BALISE SANS ATTRIBUT, SANS MANIPULATION DU CORPS	35
5.1.1.	La classe BonjourTag.....	36
5.1.2.	Le fichier exemples.tld.....	36
5.1.3.	Le fichier bonjour.jsp	37
5.1.4.	Le fichier web.xml	37
5.2.	BALISE AVEC ATTRIBUT	38
5.2.1.	La classe BonjourTag.....	38
5.2.2.	Le fichier exemples.tld.....	39
5.2.3.	Le fichier bonjour.jsp	40
5.3.	BALISE AVEC ATTRIBUT VALORISE VIA UN BEAN.....	41
5.3.1.	La classe Adherent	41
5.3.2.	Servlet qui crée le bean	42
5.3.3.	Le fichier bonjour.jsp	42
5.4.	BALISE AVEC ATTRIBUT ET DECLARANT UNE VARIABLE SCRIPT	43
5.4.1.	Le descripteur de balise.....	43
5.4.2.	La classe décrivant la variable script.....	43
5.4.3.	Le gestionnaire de balise	44
5.4.4.	La JSP.....	45
5.5.	BALISES PERSONNALISEES (VERSION 1.2).....	48
5.5.1.	Corps de balise avec variables de script.....	48
5.5.1.1.	La classe BonjourTag.....	49
5.5.1.2.	La classe BonjourTagExtraInfo	50
5.5.1.3.	Le fichier exemples.tld.....	50
5.5.1.4.	Le fichier bonjour.jsp.....	51
5.5.2.	Balise implémentant une itération (version 1.2)	52
5.5.2.1.	Servlet mettant un objet liste à disposition.....	53
5.5.2.2.	La classe BonjourListeTag.....	54
5.5.2.3.	La classe BonjourListeTagExtraInfo	56
5.5.2.4.	Le fichier bonjourListe.jsp	56
5.5.2.5.	Le fichier exemples.tld.....	57
5.5.3.	Imbrication de balises (version 1.2)	58
5.5.3.1.	Servlet qui crée la liste	59
5.5.3.2.	Modification de la classe BonjourListeTag	60
5.5.3.3.	La classe InfosAdherentTag.....	61
5.5.3.4.	La classe InfosAdherentTagExtraInfo	62
5.5.3.5.	Le fichier bonjourListe.jsp	62
5.5.3.6.	Le fichier exemples.tld.....	63
6.	UTILISATION DE BALISES JSTL.....	64
6.1.	SAISIE CONDITIONNELLE.....	64
6.1.1.	La classe « Client »	65
6.1.2.	Le servlet.....	66
6.1.3.	La JSP.....	67

6.1.4.	Le fichier web.xml	68
6.2.	BOUCLE SUR UNE LISTE	69
6.2.1.	Ajouts dans la classe « Client ».....	69
6.2.2.	Ajouts dans le servlet	70
6.2.3.	Ajouts dans la JSP	70
6.3.	IMPORT	71
6.3.1.	La page qui importe l'autre	71
6.3.2.	La page importée	71
6.4.	INTERNATIONALISATION.....	72
6.4.1.	Choix de la langue initiale.....	72
6.4.1.1.	Les fichiers .properties	73
6.4.1.2.	Le fichier web.xml	74
6.4.1.3.	Les JSP	75
6.4.2.	Choix de la langue par l'utilisateur	77
6.4.2.1.	Structure de l'application	77
6.4.2.2.	La JSP de connexion	78
6.4.2.3.	Le servlet.....	79
6.4.2.4.	Les 2 autres JSP	79

1. PREAMBULE

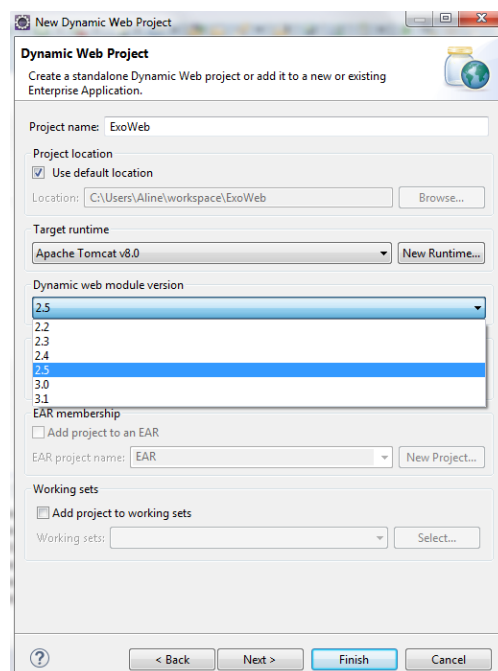
Création du projet Web dynamique

Sous Eclipse « File »>New>Dynamic Web Project, si vous n'avez pas cette option, utilisez Other (tout en bas). Dans Other, vous trouverez tous les Wizards.



Sélectionnez le répertoire Web, le déployer et choisir Dynamic Web Project.

Dans un premier temps, nous travaillons avec une version Java EE 2.5.



2. MISE EN ŒUVRE DE SERVLETS

Objectifs : Développement de servlets.

2.1. AFFICHAGE D'UN TEXTE

Objectifs: Manipulation de Servlet et de l'objet HttpServletResponse.

Un premier exercice qui affiche tout simplement un texte, en codant un servlet.

```
package tp1;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.ServletOutputStream;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Servlet implementation class ServBonjour
 */
public class ServTp1 extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public ServBonjour() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
    response) */
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        // TODO Auto-generated method stub
        aFaire(request, response);
    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
    response) */
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        // TODO Auto-generated method stub
        aFaire(request, response);
    }
}
```

```
private void aFaire(HttpServletRequest request, HttpServletResponse response)
throws IOException {

    // mise en place d'un flux de sortie
    ServletOutputStream sortie = response.getOutputStream() ;

    // entête de la page
    sortie.println("<html><title>Bonjour</title> <body>") ;

    // corps

    sortie.println("<p> Bienvenue chez Globalknowledge <br>") ;

    // fin de page
    sortie.println("</body></html>") ;
    sortie.close() ;
}

}
```

2.2. AFFICHAGE D'UN CONTENU DE BOUCLE

Exercice qui affiche tout simplement le contenu d'une variable de boucle.

```
package tp1 ;

import javax.servlet.* ;           // Où cycle de vie de toute servlet
import javax.servlet.http.* ;     // Pour les servlets HTTP
import java.io.* ;                // Pour le flux en sortie

public class AfficheBoucle extends HttpServlet
{
    /* Constructeur nécessaire pour certains serveurs */
    public AfficheBoucle() {
        super() ;
    }
    /* Méthodes doGet et doPost à définir comme précédemment
    .../...
    /* Méthode service, lancée pour chaque requête client */
    public void aFaire(HttpServletRequest request,
                        HttpServletResponse response)
                        throws ServletException, IOException {

        // mise en place d'un flux de sortie
        ServletOutputStream sortie = response.getOutputStream() ;

        // entête de la page
        sortie.println("<html><body>") ;
        sortie.println("<H1> Début </H1>") ;
        // corps
        for ( int i = 1 ; i < 11 ; i++ )
        {
            sortie.println("<p> i = " + i + "<br>") ;
        }
        // fin de page
        sortie.println("</body></html>") ;
        sortie.close() ;
    }
}
```


2.3. SAISIE D'UN FORMULAIRE

Afficher un formulaire permettant la saisie du titre et de la catégorie d'un livre.
Une fois que l'utilisateur a saisi son formulaire, un autre servlet affiche les informations saisies.

```
package tp2;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

public class FormLivre extends HttpServlet {
    public FormLivre () {
        super();
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
    response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        // TODO Auto-generated method stub
        aFaire(request, response);
    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
    response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        // TODO Auto-generated method stub
        aFaire(request, response);
    }
}
```

```

public void aFaire(HttpServletRequest requete,
                  HttpServletResponse reponse)
                  throws ServletException, IOException {

    PrintWriter sortie = reponse.getWriter() ;
    reponse.setContentType("text/html");

    //Génération de la page html contenant le formulaire
    sortie.println("<html><head><title> Livre </title>");
    sortie.println("</head><body> <h1>Ouvrage demandé</h1> ");
    sortie.println("<form method='post'
                  action='CtlLivre'>");
    sortie.println("<p> Quel est son titre ? </p>");
    sortie.println("<input name=titre type=text size=12/>");
    sortie.println("<p> Quelle est sa catégorie ? </p>");
    sortie.println("<input name=categorie type=text size=12/>");
    sortie.println("<input type=submit name='Valider'>");
    sortie.println("</form></body></html>");

    sortie.close();
}
}

```

Version « Sélection d'une catégorie »

```

private void aFaire(HttpServletRequest request, HttpServletResponse response) throws
IOException {

    // mise en place d'un flux de sortie
    ServletOutputStream sortie = response.getOutputStream() ;

    //Génération de la page html contenant le formulaire
    sortie.println("<html><head><title>Livre</title>");
    sortie.println("</head><body> <h1>Ouvrage demandé</h1>");
    sortie.println("<form method='post' action='CtlLivre'>");
    sortie.println("<p> Quel est son titre ? </p>");
    sortie.println("<input name=titre type=text size=25/>");
    sortie.println("<p> Quelle est sa catégorie ? </p>");
    sortie.println("<select name='categorie'>" +
    " <option>Thriller</option>" +
    " <option>Roman</option>" +
    " <option>Bande dessin&eacute;e</option>" +
    "<option>Histoire</option>" +
    "</select>" );
    sortie.println("<input type=submit name=Valider/>");
    sortie.println("</form></body></html>");

    sortie.close();
}
}

```

Récupération de la saisie et affichage

```
package tp2; /* Récupération des paramètres du formulaire */

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.util.Enumeration;

public class CtlLivre extends HttpServlet {

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        aFaire (request,response);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        aFaire (request,response);
    }
    private void faire(HttpServletRequest requete,
                        HttpServletResponse reponse)
                        throws ServletException, IOException {

        //Variables titre et categorie
        String titre;
        String categorie;
        //récupération des valeurs saisies par l'utilisateur
        titre=request.getParameter("titre");
        categorie=request.getParameter("categorie");

        // mise en place d'un flux de sortie
        ServletOutputStream sortie = response.getOutputStream() ;

        //Génération de la page html contenant le formulaire
        sortie.println("<html><head><title>Livre</title>");
        sortie.println("</head><body> <h1>Ouvrage saisi</h1>");

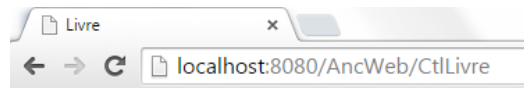
        sortie.println("<p> Quel est son titre : </p>" + titre);

        sortie.println("<p> Quelle est sa catégorie : </p>" + categorie);

        sortie.println("</body></html>");

        sortie.close();
    }
}
```

Supplément Récupération de la saisie avec une Enumeration – 3 paramètres



Ouvrage saisi

Quel est son titre :

Titi

Quelle est sa catégorie :

Bande dessinée

Les paramètres de la page :

nom du paramètre : titre valeur : Titi

nom du paramètre : categorie valeur : Bande dessinée

nom du paramètre : Valider valeur : Valider

```
private void faire(HttpServletRequest requete,
    HttpServletResponse reponse) throws ServletException, IOException {
//Variables titre et categorie
    String titre;
    String categorie;
//Récup de tous les parametres de la page
    String nomParam, valeurParam ;
    Enumeration e = request.getParameterNames();

//récupération des valeurs saisies par l'utilisateur
    titre=request.getParameter("titre");
    categorie=request.getParameter("categorie");

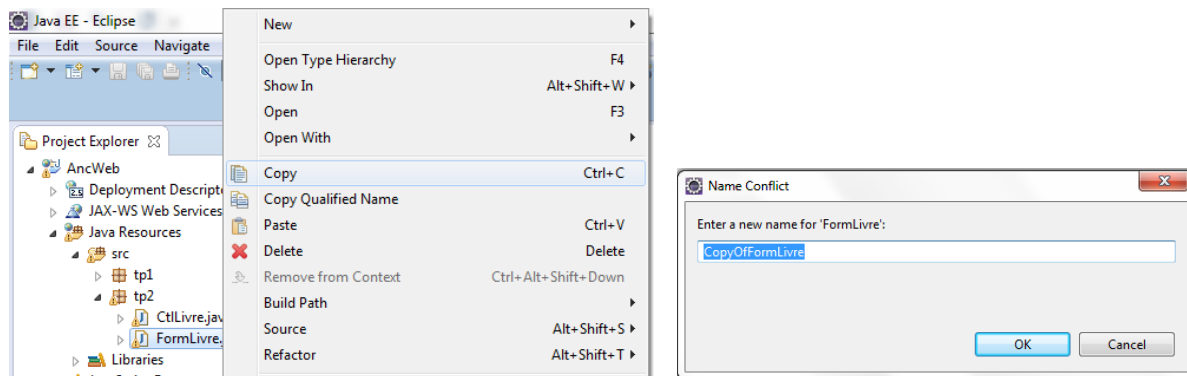
    // mise en place d'un flux de sortie
    ServletOutputStream sortie = response.getOutputStream() ;
    //Génération de la page html contenant le formulaire
    sortie.println("<html><head><title>Livre</title>");
    sortie.println("</head><body> <h1>Ouvrage saisi</h1>");
    sortie.println("<p> Quel est son titre : </p>" + titre);
    sortie.println("<p> Quelle est sa catégorie : </p>" + categorie);
    sortie.println("<p><h3> Les paramètres de la page : </h3></p>");
    while ( e.hasMoreElements() )
    {
        nomParam = (String) e.nextElement();
        valeurParam = request.getParameter(nomParam);
        sortie.println("<p> nom du paramètre : " + nomParam + " ");
        sortie.println("valeur : " + valeurParam + " </p>");
    }
    sortie.println("</body></html>");
    sortie.close();
}
```

2.4. MANIPULATION DU WEB.XML

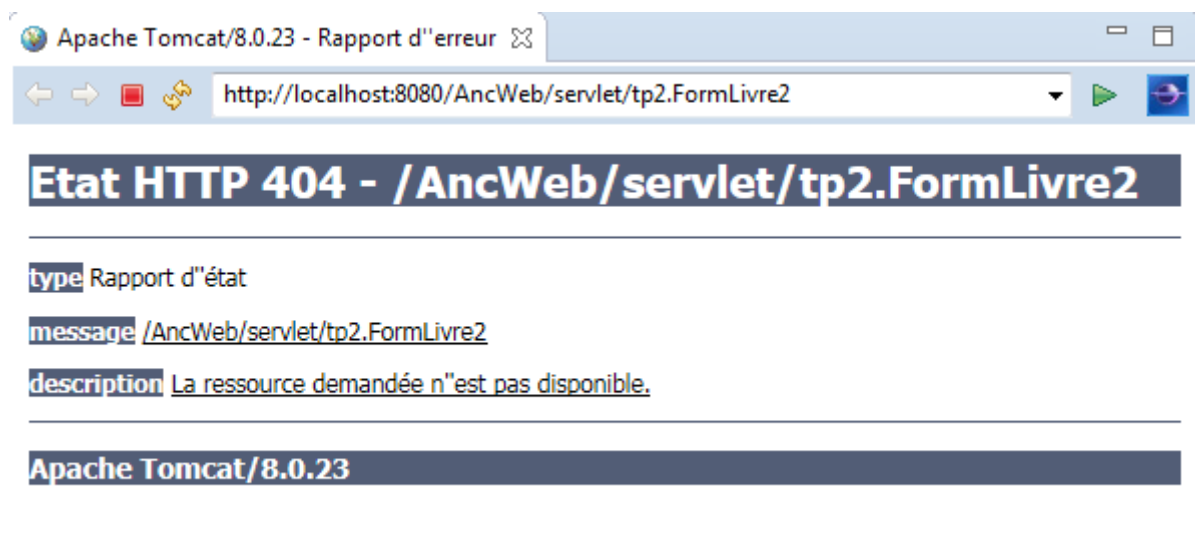
Objectifs: Manipulation du descripteur de déploiement et de l'objet ServletContext.

2.4.1. COPIER/COLLER DU SERVLET

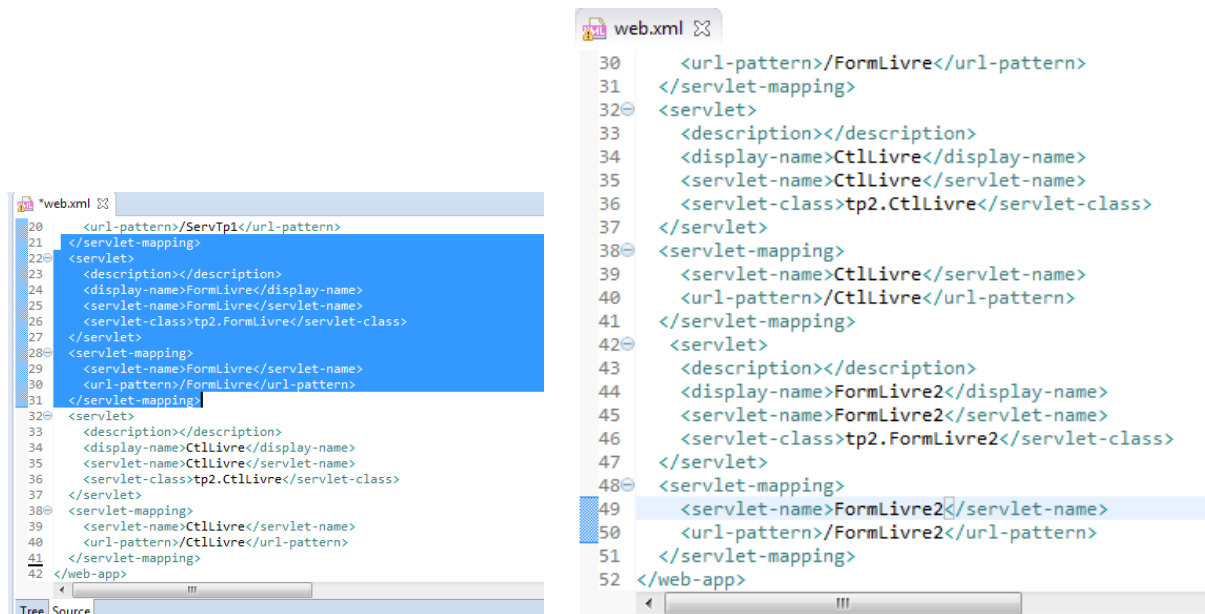
Sélectionner le Servlet, click droit « Copy ». Sélectionner le package et faire « Paste », changer le nom de la copie



Exécuter le servlet : on obtient une erreur 404 correspond à une ressource non trouvée dans l'environnement d'exécution.



Ouvrir le web.xml, copier les balises `<servlet>` et `<servlet-mapping>` de FormLivre et les coller à la fin de l'ensemble des balises « servlet ».



```

web.xml
20 <url-pattern>/ServTp1</url-pattern>
21 </servlet-mapping>
22 <servlet>
23 <description></description>
24 <display-name>FormLivre</display-name>
25 <servlet-name>FormLivre</servlet-name>
26 <servlet-class>tp2.FormLivre</servlet-class>
27 </servlet>
28 <servlet-mapping>
29 <servlet-name>FormLivre</servlet-name>
30 <url-pattern>/FormLivre</url-pattern>
31 </servlet-mapping>
32 <servlet>
33 <description></description>
34 <display-name>CtlLivre</display-name>
35 <servlet-name>CtlLivre</servlet-name>
36 <servlet-class>tp2.CtlLivre</servlet-class>
37 </servlet>
38 <servlet-mapping>
39 <servlet-name>CtlLivre</servlet-name>
40 <url-pattern>/CtlLivre</url-pattern>
41 </servlet-mapping>
42 <servlet>
43 <description></description>
44 <display-name>FormLivre2</display-name>
45 <servlet-name>FormLivre2</servlet-name>
46 <servlet-class>tp2.FormLivre2</servlet-class>
47 </servlet>
48 <servlet-mapping>
49 <servlet-name>FormLivre2</servlet-name>
50 <url-pattern>/FormLivre2</url-pattern>
51 </servlet-mapping>
52 </web-app>

```

2.4.2. EXECUTION DU SERVLET 2



web.xml Livres

http://localhost:8080/AncWeb/FormLivre2

Ouvrage demandé

Quel est son titre ?

Quelle est sa catégorie ?

Thriller

Soumettre la requête

2.4.3. AJOUT PARAMETRE D'APPLICATION FICHIER WEB.XML



```

web.xml
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" :
3 <display-name>AncWeb</display-name>
4 <welcome-file-list>
5 <welcome-file>index.html</welcome-file>
6 <welcome-file>index.htm</welcome-file>
7 <welcome-file>index.jsp</welcome-file>
8 <welcome-file>default.html</welcome-file>
9 <welcome-file>default.htm</welcome-file>
10 <welcome-file>default.jsp</welcome-file>
11 </welcome-file-list>
12 <context-param>
13 <param-name>societe</param-name>
14 <param-value>GlobalKnowledge</param-value>
15 </context-param>
16
17 <servlet>
18 <description></description>

```

Paramètres de l'appli

Servlet FormLivre2

```

package tp2;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.ServletOutputStream;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class FormLivre2 extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public FormLivre2() { super(); }
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException { aFaire(request, response);}
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException { aFaire(request, response);}
    private void aFaire(HttpServletRequest request, HttpServletResponse response)
throws IOException {

        // Récupération du contexte d'exécution de l'application
        ServletContext ctx = getServletContext();
        //Récupération de la valeur du paramètre d'application
        String societe = ctx.getInitParameter("societe");

        // mise en place d'un flux de sortie
        ServletOutputStream sortie = response.getOutputStream() ;

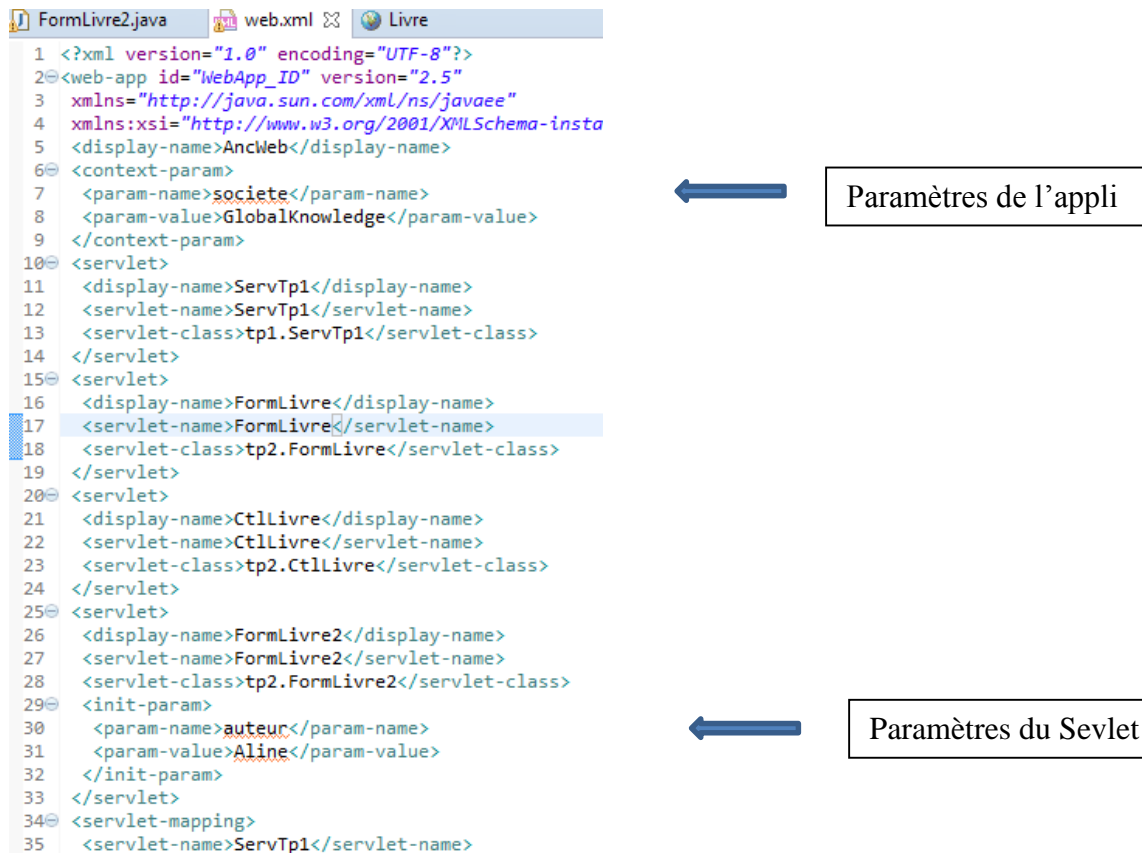
        //Génération de la page html contenant le formulaire
        sortie.println("<html><head><title>Livre</title>");
        sortie.println("</head><body> <h1>Ouvrage demandé</h1>");
        sortie.println("<h2> chez " + societe + "</h2>");

        sortie.println("<form method='post' action='CtlLivre'>");
        sortie.println("<p> Quel est son titre ? </p>");
        sortie.println("<input name=titre type=text size=25/>");
        sortie.println("<p> Quelle est sa catégorie ? </p>");
        sortie.println("<select name='categorie'>" +
            " <option>Thriller</option>" +
            " <option>Roman</option>" +
            " <option>Bande dessin&eacute;e</option>" +
            "<option>Histoire</option>" +
            "</select>" );
        sortie.println("<input type=submit name='Valider'/>");
        sortie.println("</form></body></html>");
        sortie.close();
    }
}

```

2.4.4. PARAMETRE DU SERVLET

Le web.xml :



The screenshot shows the `web.xml` file in an IDE. The file contains XML configuration for a web application. Two blue arrows point to specific sections of the code:

- The first arrow points to the `<context-param>` section (lines 6-9), which defines application parameters like `societe` and `GlobalKnowledge`. A box labeled "Paramètres de l'appli" is next to it.
- The second arrow points to the `<init-param>` section (lines 29-32) within the `<servlet>` element for `FormLivre2`, which defines servlet parameters like `auteur` and `Aline`. A box labeled "Paramètres du Sevlet" is next to it.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app id="WebApp_ID" version="2.5"
3   xmlns="http://java.sun.com/xml/ns/javaee"
4   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5   <display-name>AncWeb</display-name>
6   <context-param>
7     <param-name>societe</param-name>
8     <param-value>GlobalKnowledge</param-value>
9   </context-param>
10  <servlet>
11    <display-name>ServTp1</display-name>
12    <servlet-name>ServTp1</servlet-name>
13    <servlet-class>tp1.ServTp1</servlet-class>
14  </servlet>
15  <servlet>
16    <display-name>FormLivre</display-name>
17    <servlet-name>FormLivre</servlet-name>
18    <servlet-class>tp2.FormLivre</servlet-class>
19  </servlet>
20  <servlet>
21    <display-name>CtlLivre</display-name>
22    <servlet-name>CtlLivre</servlet-name>
23    <servlet-class>tp2.CtlLivre</servlet-class>
24  </servlet>
25  <servlet>
26    <display-name>FormLivre2</display-name>
27    <servlet-name>FormLivre2</servlet-name>
28    <servlet-class>tp2.FormLivre2</servlet-class>
29    <init-param>
30      <param-name>auteur</param-name>
31      <param-value>Aline</param-value>
32    </init-param>
33  </servlet>
34  <servlet-mapping>
35    <servlet-name>ServTp1</servlet-name>

```

Le code Java dans un servlet :

```

// Récupération du contexte d'exécution de l'application
ServletContext ctx = getServletContext();
//Récupération de la valeur du paramètre d'application
String societe = ctx.getInitParameter("societe");
//Récup de la valeur du paramètre du servlet
String auteur = this.getInitParameter("auteur");
// mise en place d'un flux de sortie
ServletOutputStream sortie = response.getOutputStream() ;
//Génération de la page html contenant le formulaire
sortie.println("<html><head><title>Livre</title>");
sortie.println("</head><body> <h1>Ouvrage demandé</h1>");
sortie.println("<h2> chez " + societe + "</h2>");
sortie.println("<h3> auteur du servlet : " + auteur + "</h2>");
sortie.println("<form method='post' action='CtlLivre'>");
sortie.println("<p> Quel est son titre ? </p>");
sortie.println("<input name=titre type=text size=25/>");
sortie.println("<p> Quelle est sa catégorie ? </p>");
sortie.println("<select name='categorie'> +
" <option>Thriller</option> +
" <option>Roman</option> +
" <option>Bande dessin&eacute;</option>"+
"<option>Histoire</option> +
"</select> ";
sortie.println("<input type=submit name='Valider'/>");
sortie.println("</form></body></html>");
sortie.close();
}}

```


2.4.5. INFORMATIONS SUR LE SERVEUR

```

package test;
import java.io.IOException;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.ServletOutputStream;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class InfoServer extends javax.servlet.http.HttpServlet implements
    javax.servlet.Servlet {
    public void service(HttpServletRequest req, HttpServletResponse rep)
        throws ServletException, IOException {
        // Recup Contexte d'execution de l'appli
        ServletContext sc = getServletContext();
        String info = sc.getServerInfo();
        String real=sc.getRealPath("");
        String nomctx=    sc.getServletContextName();
        ServletOutputStream out = rep.getOutputStream();
        // écriture log
        sc.log("ECRIRE DANS LOG DE " + info);
        // création de la page HTML
        out.println("<HTML>");
        out.println("<HEAD><TITLE>Info Serveur</TITLE></HEAD>");
        out.println("<BODY> " +
            info + "<br>" +
            nomctx + "<br>" +
            real +
            "</BODY></HTML>");
        // toujours fermer le flux d'écriture
        out.close();
    }
}

```

Résultats en HTML

Apache Tomcat/5.5.23

MaintS

C:\Documents and Settings\acas\workspacede ne pas
effacer\.metadata\plugins\org.eclipse.wst.server.core\tmp0\webapps\MaintS

A la console

10 oct. 2008 15:38:25 org.apache.catalina.core.ApplicationContext log
INFO: ECRIRE DANS LOG DE Apache Tomcat/5.5.23

3. COMMUNICATION ENTRE SERVLETS

Objectifs : Développement de chainage de servlets, manipulation des méthodes *forward* et *include* de l'objet *RequestDispatcher* et manipulation de l'objet *HttpSession*.

3.1. GESTION DE LA NAVIGATION

Ecrire un servlet Contrôleur qui pilote l'application. Il appelle le servlet *FormLogin* qui affiche le nom et le mot de passe de l'utilisateur quand celui-ci n'est pas identifié.

Sinon, si l'utilisateur est connecté, il appelle le servlet *FormLivre*.

3.1.1. LE CONTROLEUR

```
package tp3;

import java.io.IOException;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletContext;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;
```

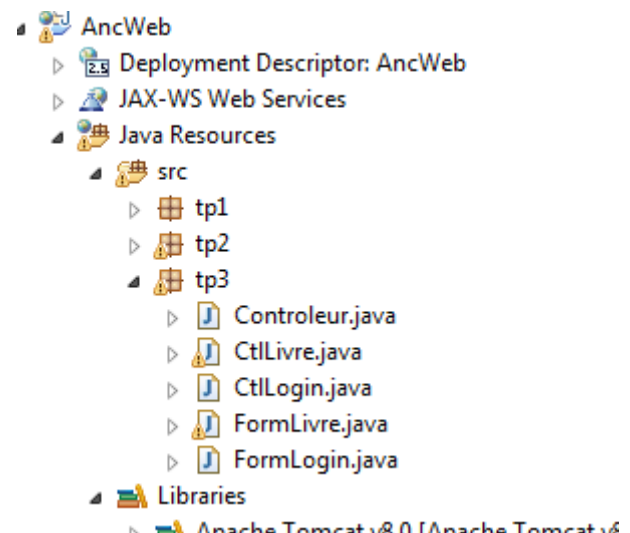
```
public class Controleur extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public Controleur() {
        super();
    }

    protected void doGet(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException
    {aFaire(request, response); }
    protected void doPost(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException
    {aFaire(request, response); }

    private void aFaire(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

        // Recupération du contexte pour effectuer du chainage de servlet
        ServletContext ctx = getServletContext();
```



```
RequestDispatcher dispatch = null;
// Initialisation du parametre de la page
String page = null;

// récupération du parametre dans la demande de l'utilisateur

page = request.getParameter("page");

// Aiguillage en fonction de la demande de l'utilisateur

if (page == null) {
    // On vient de nulle part --> Formulaire de login

    dispatch = ctx.getRequestDispatcher("/FormLogin");
    dispatch.forward(request, response);
} else {

    if ("1".equals(page)) {
        // on vient du login --> enregistrement
        dispatch = ctx.getRequestDispatcher("/CtlLogin");
        dispatch.include(request, response);
        // affichage résa
        dispatch = ctx.getRequestDispatcher("/FormLivre");
        dispatch.forward(request, response);
    } else {

        if ("2".equals(page)) {
            // on vient du livre --> enregistrement
            dispatch = ctx.getRequestDispatcher("/CtlLivre");
            dispatch.include(request, response);
            // affichage résa
            dispatch = ctx.getRequestDispatcher("/FormLivre");
            dispatch.forward(request, response);
        }
    }
}
}
```

3.1.2. LE FORMULAIRE DE LOGIN

```

package tp3;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.ServletOutputStream;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
/**
 * Servlet implementation class FormLogin
 */
public class FormLogin extends HttpServlet {
    private static final long serialVersionUID = 1L;
    public FormLogin() { super(); }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        // TODO Auto-generated method stub
        aFaire(request, response);
    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        // TODO Auto-generated method stub
        aFaire(request, response);
    }
    private void aFaire(HttpServletRequest request, HttpServletResponse response)
throws IOException {

        // mise en place d'un flux de sortie
        ServletOutputStream sortie = response.getOutputStream() ;

        //Génération de la page html contenant le formulaire
        sortie.println("<html><head><title>Login</title>");
        sortie.println("</head><body> <h1>Enregistrement</h1>");
        sortie.println("<form method='post' action='Controleur?page=1'>");
        sortie.println("<p> Quel est votre nom ? </p>");
        sortie.println("<input name=nom type=text size=25/>");
        sortie.println("<p> Votre mot de passe ? </p>");
        sortie.println("<input name=mdp type=text size=25/>");
        sortie.println("<input type=submit name='Valider'/>");
        sortie.println("</form></body></html>");

        sortie.close();
    }
}

```

3.1.3. LE FORMULAIRE DE RESERVATION

Pour les tests de navigation, il fallait juste changer l'action dans le formulaire comme suit :

```
sortie.println("<form method='post' action='Controleur?page=2'>");
```

Pour l'affichage du nom de l'utilisateur, enregistré dans la session en chaîne de caractère, voici le formulaire (seule la méthode aFaire est notée).

```
private void aFaire(HttpServletRequest request, HttpServletResponse response)
throws IOException {
    //r  cup  ration du nom de l'utilisateur dans la session
    HttpSession session = request.getSession();
    String user = (String) session.getAttribute("user");

    // mise en place d'un flux de sortie
    ServletOutputStream sortie = response.getOutputStream() ;

    //G  n  ration de la page html contenant le formulaire
    sortie.println("<html><head><title>Livres</title>");
    sortie.println("</head><body><h3>Bonjour " + user + " </h3>");
    sortie.println("<h1>Ouvrage demand   </h1>");
    sortie.println("<form method='post' action='Controleur?page=2'>");
    sortie.println("<p> Quel est son titre ? </p>");
    sortie.println("<input name=titre type=text size=25/>");
    sortie.println("<p> Quelle est sa cat  gorie ? </p>");
    sortie.println(" <select name='categorie'>" +
        " <option>Thriller</option>" +
        " <option>Roman</option>" +
        " <option>Bande dessin  e</option>" +
        "<option>Histoire</option>" +
        "</select>" );
    sortie.println("<input type=submit name='Valider' />");
    sortie.println("</form></body></html>");

    sortie.close();
}
```

3.2. GESTION DES SESSIONS

Servlet CtlLogin :

Enregistrement dans la session, en chaine de caractère, du nom de l'utilisateur (seule la méthode aFaire est notée).

```
private void aFaire(HttpServletRequest request, HttpServletResponse response)
throws IOException {
    //récupération du nom de l'utilisateur saisi dans le formulaire
    String user = request.getParameter("nom");
    //Création de la session
    HttpSession session = request.getSession(true);
    //enregistrement du user dans la session
    session.setAttribute("user", user);
}
```

3.3. LES JAVABEANS NECESSAIRES A L'APPLICATION

3.3.1. LES BEANS

Le livre

```
package mesBeans;
public class Livre {
    private String titre;
    private String categorie;
    public Livre() {
        titre="non renseigné";
        categorie="inconnue";
    }
    public Livre(String titre, String categorie) {
        this.titre = titre;
        this.categorie = categorie;
    }
    public String getTitre() {
        return titre;
    }
    public void setTitre(String titre) {
        this.titre = titre;
    }
    public String getCategorie() {
        return categorie;
    }
    public void setCategorie(String categorie) {
        this.categorie = categorie;
    }
    @Override
    public String toString() {
        return "Livre [Titre=" + getTitre() + ", Categorie="
            + getCategorie() + "]";
    }
}
```

Le client

```

package mesBeans;

import java.util.ArrayList;

public class Client {
    private String user;
    private String mdp;
    private ArrayList<Livre> livres;

    public Client() {
        user = "Inconnu";
        mdp = "vide";
        livres = new ArrayList<Livre>();
    }

    public Client(String user, String mdp) {
        this();
        this.user = user;
        this.mdp = mdp;
    }

    public String getUser() {
        return user;
    }

    public void setUser(String user) {
        this.user = user;
    }

    public String getMdp() {
        return mdp;
    }

    public void setMdp(String mdp) {
        this.mdp = mdp;
    }

    public ArrayList<Livre> getLivres() {
        return livres;
    }

    public void setLivres(ArrayList<Livre> livres) {
        this.livres = livres;
    }

    public void ajoutLivre(Livre livre){
        getLivres().add(livre);
    }

    public void ajoutLivre(String titre, String categorie){
        //Création du Livre
        Livre livre = new Livre(titre, categorie);
        //enregistrement du Livre dans la liste
        getLivres().add(livre);
    }

    @Override
    public String toString() {
        return "Client [nom=" + getUser() + ", Livres="
            + getLivres() + "]";
    }
}

```

3.3.2. LA GESTION DES BEANS DANS LES SERVLETS

CtlLogin

```
private void aFaire(HttpServletRequest request, HttpServletResponse response)
throws IOException {
    //récupération du nom et mdp de l'utilisateur saisi dans le formulaire
    String nom = request.getParameter("nom");
    String mdp = request.getParameter("mdp");
    //Création de l'objet Client en fonction des valeurs saisies
    //(la liste de Livres est automatiquement gérée par le constructeur de client
    Client user= new Client(nom, mdp);
    //Création de la session
    HttpSession session = request.getSession(true);
    //enregistrement du user dans la session
    session.setAttribute("user", user);
}
```

CtlLivre

```
private void aFaire(HttpServletRequest request, HttpServletResponse response)
throws IOException {

    //récupération des valeurs saisies par l'utilisateur
    String titre=request.getParameter("titre");
    String categorie=request.getParameter("categorie");

    //Récupération de la session et du user
    HttpSession session = request.getSession();
    //récupération de l'utilisateur dans la session
    Client client = ((Client) session.getAttribute("user"));

    //Ajout du livre dans la liste du client
    client.ajoutLivre(titre, categorie);

}
```


Formulaire de réservation

Le formulaire affiche maintenant la liste des livres réservés par l'utilisateur. Pour cela, nous mettons en œuvre une boucle sur la liste de livres. Pour chaque livre issu de cette liste, nous affichons l'objet converti en chaîne de caractère. Nous rappelons, que la liste appartient au client (objet que nous avons créé), que le client est mémorisé dans la session, que les objets Livre stockés dans la liste de type ArrayList (de java.util) ont une méthode toString définie pour afficher un objet Livre en chaîne de caractères.

```
//récupération de la session
HttpSession session = request.getSession();
//récupération de l'utilisateur dans la session
Client client = ((Client) session.getAttribute("user"));
//récupération du nom de l'utilisateur
String user = client.getUser();
//récup de sa liste de livres
ArrayList<Livre> liste = client.getLivres();
// mise en place d'un flux de sortie
ServletOutputStream sortie = response.getOutputStream() ;

//Génération de la page html contenant le formulaire
sortie.println("<html><head><title>Livre</title>");
sortie.println("</head><body><h3>Bonjour " + user + " </h3>");
sortie.println("<h1>Ouvrage demandé </h1>");
sortie.println("<form method='post' action='Controleur?page=2'>");
sortie.println("<p> Quel est son titre ? </p>");
sortie.println("<input name=titre type=text size=25/>");
sortie.println("<p> Quelle est sa catégorie ? </p>");
sortie.println("<select name='categorie'>" +
" <option>Thriller</option>" +
" <option>Roman</option>" +
" <option>Bande dessinée</option>" +
"<option>Histoire</option>" +
"</select>" );
sortie.println("<input type=submit name='Valider'/>");
sortie.println("</form>");

sortie.println("<p> Vos réservations : </p>");

for (Livre livre : liste) {
sortie.println(livre.toString());
}
sortie.println("</body></html>");
sortie.close();
}
```

4. BIBLIOTHEQUE EN LIGNE

Objectifs : Cet exercice récapitulatif permet de mettre en œuvre les différents concepts et techniques abordés dans la première partie du cours servlets-JSP :

- Servlets
- Java Server Pages
- Conception modulaire
- Utilisation du motif de conception **Modèle-Vue-Contrôleur**

4.1. REMPLACER LES SERVLETS PAR DES JSP

vLogin

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Enregistrement</title>
</head>
<body>
<form method='post' action='Controleur?page=1'>
<p> Quel est votre nom ? </p>
<input name=nom type=text size=25/>
<p> Votre mot de passe ? </p>
<input name=mdp type=text size=25/>
<input type=submit name='Valider' />
</form>
</body>
</html>
```

vLivre

Pour cette JSP, nous utilisons la balise `<jsp:useBean>`, pour récupérer le Client dans la session (sauvegardé en attribut « user ») et ensuite la balise `<jsp:getProperty>` du bean « user » pour afficher la propriété « user » qui correspond au nom du client. On aurait pu changer les noms d'attributs et propriétés pour éviter les confusions, c'est aussi pour vous apprendre de bonnes manières que l'on a laissé ce code pas facile à maintenir ☺. Nous utilisons aussi EL pour afficher la même chose, c'est bien sûr ce dernier choix qu'il faut privilégier !

Pour afficher la liste des livres réservés par le Client, nous utilisons un scriptlet, pour effectuer un traitement en boucle.

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ page import="java.util.ArrayList" %>
<%@ page import="mesBeans.Client" %>
<%@ page import="mesBeans.Livre" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html><head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Livres</title>
<jsp:useBean id="user" class="mesBeans.Client" scope="session" ></jsp:useBean>
</head>
<body>
<h3>Bonjour <jsp:getProperty property="user" name="user"/> </h3>
<h1>Ouvrage demandé </h1>
<form method='post' action='Controleur?page=2'>
<p> Quel est son titre ? </p>
<input name=titre type=text size=25/>
<p> Quelle est sa catégorie ? </p>
<select name='categorie'>
<option>Thriller</option>
<option>Roman</option>
<option>Bande dessinée</option>
<option>Histoire</option>
</select>
<input type=submit name='Valider' /> </form>
<p> Vos réservations : </p>
<%
    for (Livre livre : user.getLivres()) {
        out.println(" <p>" + livre.toString() + "</p>");
    }
%>
</body></html>
```

4.2. FONCTIONNEMENT DU CONTROLEUR

Ce TP met en œuvre les principes du modèle MVC 2. On a donc clairement dissocié les trois types de composants :

- La partie interface graphique utilisateur est assurée exclusivement par des JSP.
- Les composants métiers sont pris en charge par des classes java spécifiques à partir desquelles sont instanciés les beans manipulés par l'application.
- Enfin, cette application dispose d'un point d'entrée unique : le contrôleur. Ce contrôleur a pour vocation de vérifier le statut des sessions clientes puis d'aiguiller le client vers le traitement adéquat. Dans la mesure où ce contrôleur prend en charge les requêtes HTTP, il est implémenté par un HttpServlet.

Le Servlet Controleur

```
private void aFaire(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    // Recupération du contexte pour effectuer du chainage de servlet
    ServletContext ctx = getServletContext();
    RequestDispatcher dispatch = null;
    // Initialisation du parametre de la page
    String page = null;
    // recupération du parametre dans la demande de l'utilisateur
    page = request.getParameter("page");
    // Aiguillage en fonction de la demande de l'utilisateur
    if (page == null) {
        // On vient de nulle part --> Formulaire de login
        dispatch = ctx.getRequestDispatcher("/vLogin.jsp");
        dispatch.forward(request, response);
    } else {
        if ("1".equals(page)) {
            // on vient du login --> enregistrement
            dispatch = ctx.getRequestDispatcher("/CtlLogin");
            dispatch.include(request, response);
            // affichage résa
            dispatch = ctx.getRequestDispatcher("/vLivres.jsp");
            dispatch.forward(request, response);
        } else {
            if ("2".equals(page)) {
                // on vient du livre --> enregistrement
                dispatch = ctx.getRequestDispatcher("/CtlLivre");
                dispatch.include(request, response);
                // affichage résa
                dispatch = ctx.getRequestDispatcher("/vLivres.jsp");
                dispatch.forward(request, response);
            }
        }
    }
}
```

4.3. MODULARITE DES PAGES

Afin de faciliter les mises à jour et l'évolution ultérieure de l'IHM, les JSP sont conçues de manière modulaire. Un entête et un pied de page uniques sont incorporés dans chaque JSP. La mise à jour des fichiers entête ou pied de page se répercute automatiquement sur l'ensemble des pages.

L'entête utilise une **directive** include. Elle n'est donc actualisée qu'à chaque **compilation** de la page. Le pied de page est inséré via une **action** include. Il est actualisé à chaque **invocation** de la page.



The screenshot shows a web browser window with the address bar displaying 'localhost:8080/biblio/Controleur'. The page title is 'Enregistrement'. The main content area features the heading 'Bibliothèque en ligne' in a large, bold, serif font. Below the heading, there is a horizontal line. The form contains two labels: 'Quel est votre nom ?' and 'Votre mot de passe ?'. Each label is followed by a text input field. To the right of the password input field is a button labeled 'Valider'. Below the form, there is another horizontal line, and at the bottom, the text 'Mise à jour le lundi 20 juillet 2015 - Global Knowledge' is displayed.

4.3.1. EXEMPLE DE CODE

4.3.2. ENTETE ET PIED DE PAGE

4.3.2.1. Entête

L'entête type pour chaque page générée par une JSP est constitué de quelques balises HTML. Le document ne contient aucun élément dynamique, il doit avoir une extension *.html*. Il est donc inséré dans la JSP via une **directive** include.

vEntete.html

```
<br>
<h1 align="center">Bibliothèque en ligne</h1>
<br>
<hr>
```

4.3.2.2. Pied de page

Le pied de page contient une expression évaluée dynamiquement côté serveur, il doit donc avoir une extension *.jsp*. On souhaite que cette inclusion soit ré-évaluée à chaque appel de la page, on utilise donc une **action** include.

vPiedPage.jsp

```
<%@page    language="java"
           import="java.util.Date, java.text.DateFormat" %>

<br>
<hr>
<br>
Mise à jour le
<%= DateFormat.getDateInstance(DateFormat.FULL).format(new Date()) %>
- Global Knowledge
<br>
```

4.3.3. SERVLET CONTROLEUR

Reste inchangé par rapport au point précédent.

4.3.4. FORMULAIRES

Ajouter dans les formulaires les lignes suivantes :

Juste derrière la balise ouvrante `<body>` :

```
<body>

<%@page language="java" errorPage="erreur.jsp"%>
<%@include file="entete.html" %>
```

→ Le formulaire reste le même

Juste avant la balise fermante `</body>` :

```
<jsp:include page="piedpage.jsp" flush="true"/>

</body>
</html>
```

4.3.5. FORMULAIRE DE RESERVATION DE LIVRES

A cette JSP, nous ajoutons, un lien <A HREF> pour pouvoir gérer la déconnexion du client, cela nous obligera à modifier notre Contrôleur pour traiter la nouvelle valeur du paramètre « page ». Bien sûr cette nouvelle fonctionnalité, nous oblige aussi à créer un nouveau sous-contrôleur pour gérer la déconnexion et une nouvelle JSP pour dire au revoir au client.

```
<A HREF='Contrôleur?page=3'> Se déconnecter...</A>
```

4.3.6. PAGE DE FIN DE SESSION

Pensez à mettre à jour l'attribut « session » de directive de page à « false » afin d'éviter la création d'une nouvelle session par défaut.

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ page session="false" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Fin</title>
</head>
<body>

<%@include file="vEntete.html" %>

<h3>Au revoir ${requestScope.user} </h3>
    <h1>Déconnexion </h1>

    <p> Vos réservations ont été supprimées </p>

<a href="Contrôleur">Se connecter à nouveau...</a>
<jsp:include page="vPiedPage.jsp" flush="true"/>
</body>
</html>
```


4.3.7. PAGE D'ERREUR

Cette page s'affiche si une erreur est rencontrée lors de l'exécution de certaines Jsp ou si aucune action n'est définie.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<%@page language="java" isErrorPage="true" %>
<%@page import="java.io.PrintWriter" %>

<html>
<head><title>Erreur</title></head>
<body>

<%@include file="vEntete.html" %>

<H1>Erreur </H1>

<H4>
  Une erreur est survenue durant le traitement de votre demande:
  <%= HttpUtils.getRequestURL(request) %>
</H4>

<B>Info sur l'erreur </B>
<%
  if (exception!=null) {
    out.println("message de l'erreur " + exception.getMessage() );
    PrintWriter outstream = new PrintWriter(out);
    exception.printStackTrace(outstream);
  }
  %>

<jsp:include page="vPiedPage.jsp" flush="true"/>
```

4.3.8. CLASSE LIVRE

La classe Livre implémente l'interface `java.io.Serializable` (si l'objet se trouve sur un autre serveur que le serveur web) et redéfinit la méthode `equals` (si l'on souhaite comparer des objets Livre entre eux).

```
package biblio;

public class Livre implements Java.io.Serializable {
    private String titre;
    private String categorie;

    //Constructeur
    public Livre(String titre, String categorie) {
        titre = titre;
        categorie = categorie;
    }

    //Getters et Setters
    public String getCategorie() {
        return categorie;
    }

    public String getTitre() {
        return titre;
    }

    public void setCategorie(String newCategorie) {
        categorie = newCategorie;
    }

    public void setTitre(String newTitre) {
        titre = newTitre;
    }

    public boolean equals(Object obj)
    {
        if (this == obj)
            return true;
        if (obj instanceof Livre)
        {
            Livre autreLivre = (Livre) obj;
            if (this.getTitre().equals(autreLivre.getTitre())
                && this.getCategorie().equals(autreLivre.getCategorie()))
                return true;
        }
        return false;
    }
}
```

4.3.9. CLASSE CLIENT

La classe Client reste inchangée, si ce n'est l'implémentation de l'interface `java.io.Serializable`, on pourrait aussi envisager de changer l'attribut `user` (qui n'était pas facile à maintenir) en `nomAdherent`, on peut aussi ne pas avoir un constructeur à vide, voici un autre exemple du bean Client.

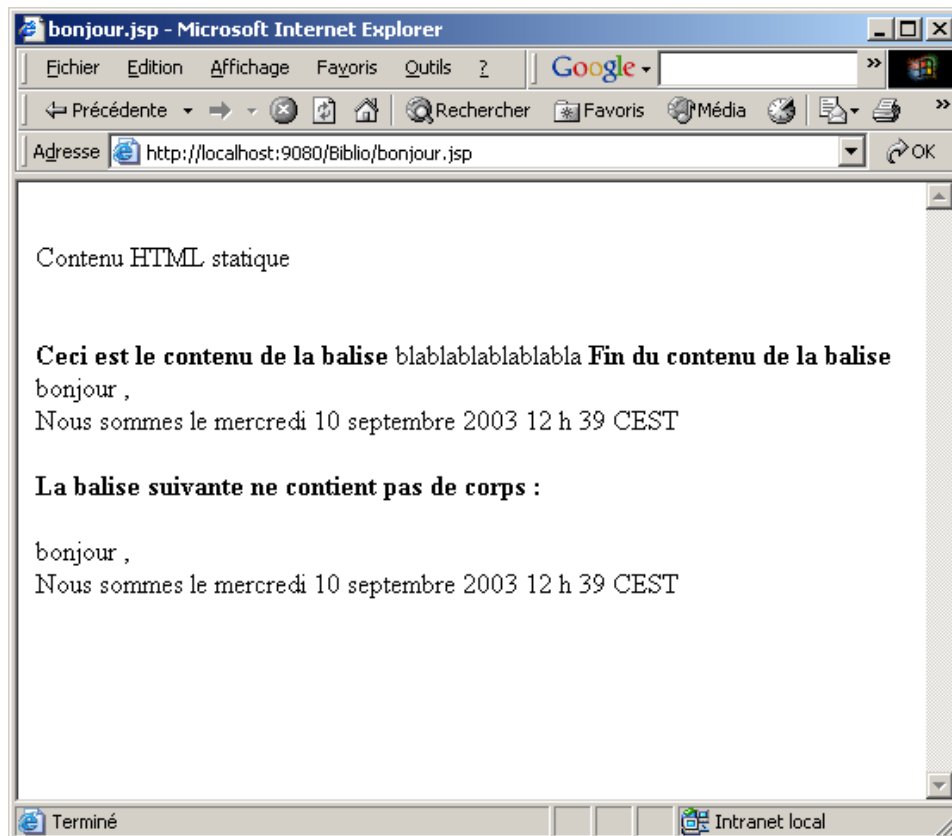
```
package biblio;
import java.util.ArrayList;
import java.io.Serializable;

public class Client implements Serializable {
    private String nomAdherent;
    private String motDePasse;
    private ArrayList listeLivres;

    public Adherent(String nom, String password) {
        super();
        nomAdherent = nom;
        motDePasse = password;
        listeLivres = new ArrayList();
    }
    public ArrayList getListeLivres() {
        return listeLivres;
    }
    public String getNom(){
        return nomAdherent;
    }
    public String getPassword(){
        return motDePasse;
    }
    public void setListeLivres(ArrayList liste) {
        listeLivres = liste;
    }
    public void setNom(String nom) {
        nomAdherent = nom;
    }
    public void setPassword(String password) {
        motDePasse = password;
    }
    public void ajoutLivre(Livre unLivre) {
        this.listeLivres.add(unLivre);
    }
    public void retraitLivre(Livre unLivre) {
        this.listeLivres.remove(unLivre);
    }
}
```

5. REALISATION DE BALISES PERSONNALISEES

5.1. BALISE SANS ATTRIBUT, SANS MANIPULATION DU CORPS



5.1.1. LA CLASSE BONJOURTAG

```
package testjsp.simple;
import java.io.IOException;
import java.util.Date;
import java.text.DateFormat;
import javax.servlet.jsp.JspException;
import javax.servlet.jsp.tagext.TagSupport;

public class BonjourTag extends TagSupport {
    private String nom;

    //Méthodes héritées de TagSupport
    public int doStartTag() throws JspException {
        return EVAL_BODY_INCLUDE;
    }
    public int doEndTag() throws JspException {
        String dt =
            DateFormat.getDateInstance().format(new Date());
        try {
            pageContext.getOut().write("bonjour <br>");
            pageContext.getOut().write("Nous sommes le "
                                     + dt + "<br>");
        }
        catch(IOException e) {
            throw new JspException("Erreur Balise");
        }
        return EVAL_PAGE;
    }
}
```

5.1.2. LE FICHIER EXEMPLES.TLD

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE taglib PUBLIC "-//Sun Microsystems,
    Inc.//DTD JSP Tag Library 1.1//EN"
    "http://java.sun.com/j2ee/dtds/web-jsptaglibrary_1_1.dtd">
<taglib>
    <tlibversion>1.0</tlibversion>
    <jspversion>1.1</jspversion>
    <shortname>ex</shortname>
    <info>Exemples de bibliothèque de balises</info>
    <tag>
        <name>Bonjour</name>
        <tagclass>testjsp.simple.BonjourTag</tagclass>
        <bodycontent>JSP</bodycontent>
        <info>Exemple de balise simple</info>
    </tag>
</taglib>
```

5.1.3. LE FICHIER BONJOUR.JSP

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<HTML>
  <HEAD>
    <META name="GENERATOR" content="IBM WebSphere Studio">
    <TITLE>bonjour.jsp</TITLE>
    <%@taglib uri="/exemples" prefix="ex"%>
  </HEAD>
  <BODY>
    <P><BR><BR>
    </P>
    Contenu HTML statique

    <ex:Bonjour>
      <h4>Début du corps de balise</h4>
      blablablabla
      <h4>Fin du corps de balise</h4>
    </ex:Bonjour>
    <br>
    <h4>La balise suivante ne contient pas de corps :</h4>
    <ex:Bonjour/>
  </BODY>
</HTML>
```

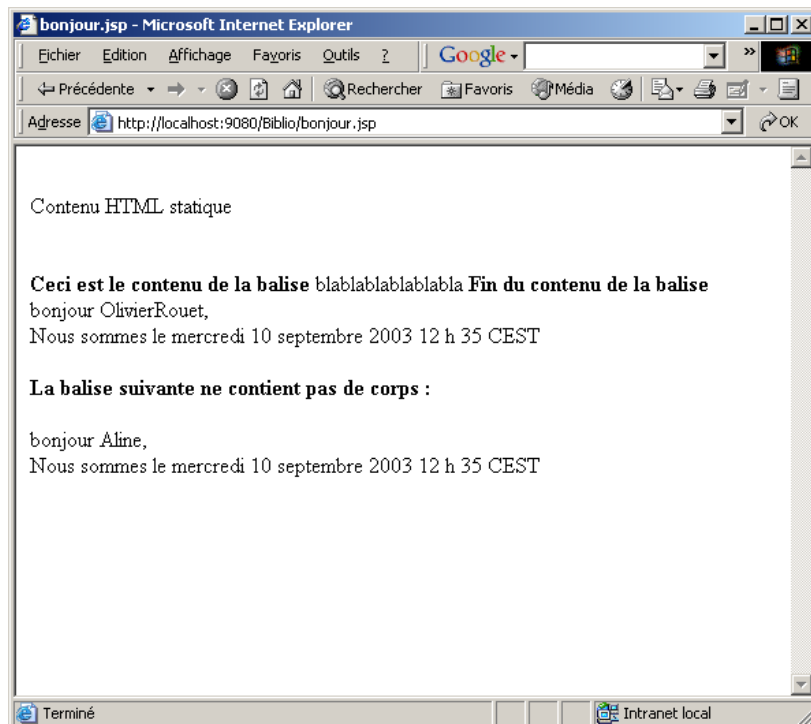
5.1.4. LE FICHIER WEB.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
"http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">
<web-app id="WebApp">
  <display-name>Biblio</display-name>
  <servlet>
    <servlet-name>ControlLogin</servlet-name>
    <display-name>ControlLogin</display-name>
    <servlet-class>control.ControlLogin</servlet-class>
  </servlet>

  [...]

  <taglib>
    <taglib-uri>/exemples</taglib-uri>
    <taglib-location>/WEB-INF/tlds/exmeples.tld</taglib-location>
  </taglib>
</web-app>
```

5.2. BALISE AVEC ATTRIBUT



5.2.1. LA CLASSE BONJOURTAG

```
package testjsp.simple;

import java.io.IOException;
import java.util.Date;
import java.text.DateFormat;
import javax.servlet.jsp.JspException;
import javax.servlet.jsp.tagext.TagSupport;

public class BonjourTag extends TagSupport {
    //Initialisation de nom obligatoire car l'attribut est facultatif
    private String nom="";
    private String prenom;

    //getters et setters pour les attributs d'instance
    public String getNom() {
        return nom;
    }
    public void setNom(String nom) {
        this.nom = nom;
    }
    public String getPrenom() {
        return prenom;
    }
    public void setPrenom(String prenom) {
        this.prenom = prenom;
    }
}
```

```
//Méthodes héritées de TagSupport
public int doStartTag() throws JspException {
    return EVAL_BODY_INCLUDE;
}
public int doEndTag() throws JspException {
    String dt =
        DateFormat.getDateInstance().format(new Date());
    try {
        pageContext.getOut().write("bonjour "
                                   + prenom + nom + "<br>");
        pageContext.getOut().write("Nous sommes le "
                                   + dt + "<br>");
    } catch(IOException e) {
        throw new JspException("Erreur Balise");
    }
    return EVAL_PAGE;
}
}
```

5.2.2. LE FICHIER EXEMPLES.TLD

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE taglib PUBLIC "-//Sun Microsystems,
    Inc.//DTD JSP Tag Library 1.1//EN"
    "http://java.sun.com/j2ee/dtds/web-jsptaglibrary_1_1.dtd">
<taglib>
    <tlibversion>1.0</tlibversion>
    <jspversion>1.1</jspversion>
    <shortname>ex</shortname>
    <info>Exemples de bibliothèque de balises</info>
    <tag>
        <name>Bonjour</name>
        <tagclass>testjsp.simple.BonjourTag</tagclass>
        <bodycontent>JSP</bodycontent>
        <info>Exemple de balise simple</info>
        <attribute>
            <name>prenom</name>
            <required>true</required>
            <rtexprvalue>true</rtexprvalue>
        </attribute>
        <attribute>
            <name>nom</name>
            <required>false</required>
            <rtexprvalue>true</rtexprvalue>
        </attribute>
    </tag>
</taglib>
```


5.2.3. LE FICHIER BONJOUR.JSP

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<%@taglib uri="/exemples" prefix="ex"%>
<HTML>
  <HEAD>
    <META name="GENERATOR" content="IBM WebSphere Studio">
    <TITLE>bonjour.jsp</TITLE>
  </HEAD>
  <BODY>
    <P><BR>
      Contenu HTML statique
    <BR>
  </P>
    <ex:Bonjour prenom="Olivier" nom="Rouet">
      <br><b>Ceci est le contenu de la balise</b>
      blablablablabla
      <b>Fin du contenu de la balise</b><br>
    </ex:Bonjour>
    <br>
    <h4>La balise suivante ne contient pas de corps :</h4>
    <ex:Bonjour prenom="Aline"/>
  </BODY>
</HTML>
```

5.3. BALISE AVEC ATTRIBUT VALORISE VIA UN BEAN

5.3.1. LA CLASSE ADHERENT

```
package testjsp.simple;

public class Adherent implements java.io.Serializable{
    private String nom;
    private String prenom;
    private String ville;
    private String email;

    //Constructeur
    public Adherent(String nom,String prenom,String ville,String email) {
        this.nom = nom;
        this.prenom = prenom;
        this.ville = ville;
        this.email = email;
    }

    //getters et setters pour les attributs d'instance
    public String getEmail() {
        return email;
    }
    public String getNom() {
        return nom;
    }
    public String getPrenom() {
        return prenom;
    }
    public String getVille() {
        return ville;
    }
    public void setEmail(String email) {
        this.email = email;
    }
    public void setNom(String nom) {
        this.nom = nom;
    }
    public void setPrenom(String prenom) {
        this.prenom = prenom;
    }
    public void setVille(String ville) {
        this.ville = ville;
    }
}
```

5.3.2. SERVLET QUI CREE LE BEAN

```

package testjsp.simple;
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class ServletBonjourTag extends HttpServlet {
    protected void service(HttpServletRequest req, HttpServletResponse rep)
        throws ServletException, IOException {
        RequestDispatcher rd;
        ServletContext sc = getServletContext();
        Adherent adh = new Adherent("Dupont", "Robert", "Paris",
                                    "bob@globalknowledge.fr");

        req.setAttribute("adherent", adh);
        rd = sc.getRequestDispatcher("/bonjour.jsp");
        rd.forward(req, rep);
    }
}

```

5.3.3. LE FICHIER BONJOUR.JSP

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<%@taglib uri="/exemples" prefix="ex"%>
<jsp:useBean id="adherent" class="testjsp.simple.Adherent" scope="request"/>

<HTML>
    <HEAD>
        <TITLE>bonjour.jsp</TITLE>
    </HEAD>
    <BODY>
        <P><BR>Contenu HTML statique<BR></P>
        <P><B>Utilisation de l'action getProperty pour récupérer le
            nom et le prénom :</B><BR>
            <jsp:getProperty name="adherent" property="prenom"/>
            <jsp:getProperty name="adherent" property="nom"/>
        </P>
        <P><B>Récupération des valeurs par scriptlet dans les attributs
            de balise :</B>
            <ex:Bonjour prenom="<%=adherent.getPrenom()%>"
                nom="<%=adherent.getNom()%>">
                <BR><B>Ceci est le contenu de la balise</B>
                    blablablablabla
                <B>Fin du contenu de la balise</B><BR>
            </ex:Bonjour>
        </P>
    </BODY>
</HTML>

```

5.4. BALISE AVEC ATTRIBUT ET DECLARANT UNE VARIABLE SCRIPT

5.4.1. LE DESCRIPTEUR DE BALISE

```
.../...  
<tag>  
  <description>  
    Un tag itérateur sur ArrayList  
  </description>  
  <name>boucle</name>  
  <tag-class>fr.logica.biblio.balise.iterate.BoucleTag</tag-class>  
  <tei-class>fr.logica.biblio.balise.iterate.BoucleTEI</tei-class>  
  <body-content>scriptless</body-content>  
  <attribute>  
    <name>liste</name>  
    <required>false</required>  
    <rtexprvalue>true</rtexprvalue>  
  </attribute>  
</tag>  
.../...
```

5.4.2. LA CLASSE DECRIVANT LA VARIABLE SCRIPT

```
package fr.logica.biblio.balise.iterate;  
  
import javax.servlet.jsp.tagext.*;  
  
public class BoucleTEI extends TagExtraInfo {  
  @Override  
  public VariableInfo[] getVariableInfo(TagData data) {  
    //création de la variable script  
    VariableInfo nbElements = new VariableInfo("nbElements",  
        "java.lang.Integer", true, VariableInfo.AT_BEGIN);  
    //création du tableau à retourner  
    VariableInfo[] info = new VariableInfo[1];  
    //Valorisation du tableau  
    info[0] = nbElements;  
    return info;  
  }  
}
```

5.4.3. LE GESTIONNAIRE DE BALISE

```
package fr.logica.biblio.balise.iterate;
import javax.servlet.jsp.*;
import javax.servlet.jsp.tagext.*;
import java.io.IOException;
import java.util.ArrayList;

public class BoucleTag extends SimpleTagSupport {
    private ArrayList liste; //l'attribut
    private int nbElements;

    public BoucleTag() {super();}

    //Getter/setter obligatoire pour les attributs
    public ArrayList getListe() { return liste;}

    public void setListe(ArrayList liste) {this.liste = liste;}

    public void doTag() throws JspException, IOException {
        // Récup du contexte de la jsp
        JspContext jsp = getJspContext();
        // Récup taille de la liste
        if (getListe() == null) {
            nbElements = 0;
        } else {
            nbElements = getListe().size();
        }
        // màj du contexte Pour la mise à dispo de la variable nbElements
        jsp.setAttribute("nbElements", nbElements);

        //Affichage du corps tel que
        getJspBody().invoke(null);
    }
}
```

5.4.4. LA JSP

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib uri="/WEB-INF/tlds/taglib.tld" prefix="tld" %>
<%@ page import="java.util.ArrayList" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html><head><title>Test indép</title></head>
<body>
<% ArrayList col = new ArrayList();
col.add("uno");
col.add("dos");
col.add("tres");%>
```

Contenu : <%=col.size() %>

1) la balise n'a pas l'attribut liste

```
<tld:boucle>gestion interne du corps variable script ${nbElements} </tld:boucle><br>
nb éléments = ${nbElements} <br>
```

2) la balise a l'attribut liste en EL

```
<tld:boucle liste="${col}"> gestion interne du corps variable script ${nbElements} </tld:boucle><br>
nb éléments = ${nbElements} <br>
```

3) la balise a l'attribut liste en balise jsp

```
<tld:boucle liste="<%=col %>">gestion interne du corps variable script ${nbElements}
</tld:boucle><br>
nb éléments = ${nbElements}
</body></html>
```

Résultats :

Contenu : 3

1) la balise n'a pas l'attribut liste gestion interne du corps variable script 0

nb éléments = 0

2) la balise a l'attribut liste en EL gestion interne du corps variable script 0

nb éléments = 0

3) la balise a l'attribut liste en balise jsp gestion interne du corps variable script 3

nb éléments = 3

Remarque 1:

Si dans le gestionnaire de balise on met en commentaire la ligne :

```
//Affichage du corps tel que
getJspBody().invoke(null);
```

Le résultat est le suivant :

Contenu : 3

1) la balise n'a pas l'attribut liste

nb éléments = 0

2) la balise a l'attribut liste en EL

nb éléments = 0

3) la balise a l'attribut liste en balise jsp

nb éléments = 3

Remarque 2:

On modifie le gestionnaire :

A chaque itération, on récupère le contenu corps de la balise,

Si la liste est vide, on ne récupère pas le contenu du corps,

On renseigne la variable script qu'une fois en fin de gestion de balise

Voici le code

```
public void doTag() throws JspException, IOException {
    // Récup du contexte de la jsp
    JspContext jsp = getJspContext();
    // Récup taille de la liste
    if (getListe() == null) {
        nbElements = 0;
    } else {
        nbElements = getListe().size();
        // Manip du corps à chaque itération
        ArrayList<String> maListe = getListe();
        for (String element : maListe) {
            //Affichage du corps tel que
            getJspBody().invoke(null);
        }
    }
    // mäj du contexte Pour la mise à dispo de la variable nbElements
    jsp.setAttribute("nbElements", nbElements);
}
```

Le résultat de la JSP est le suivant :

Contenu : 3

1) la balise n'a pas l'attribut liste

nb éléments = 0

2) la balise a l'attribut liste en EL

nb éléments = 0

3) la balise a l'attribut liste en balise jsp gestion interne du corps variable script 0
gestion interne du corps variable script 0 gestion interne du corps variable script 0
nb éléments = 3

Si l'on souhaite le contenu de la variable script à chaque itération, il faut renseigner le jspContext à chaque itération !

Voici le code

Afin d'obtenir la variable script à chaque itération

```
public void doTag() throws JspException, IOException {
    // Récup du contexte de la jsp
    JspContext jsp = getJspContext();
    // Récup taille de la liste
    if (getListe() == null) {
        nbElements = 0;
    } else {
        // Manip du corps à chaque itération

        for (int i = 0; i < getListe().size(); i++) {

            // màj du contexte Pour la mise à dispo de la variable nbElements
            jsp.setAttribute("nbElements", i);
            //Affichage du corps tel que
            getJspBody().invoke(null);
        }
        nbElements = getListe().size();
    }
    // màj du contexte Pour la mise à dispo de la variable nbElements
    jsp.setAttribute("nbElements", nbElements);
}
```

Le résultat de la JSP est le suivant :

Contenu : 3

1) la balise n'a pas l'attribut liste

nb éléments = 0

2) la balise a l'attribut liste en EL

nb éléments = 0

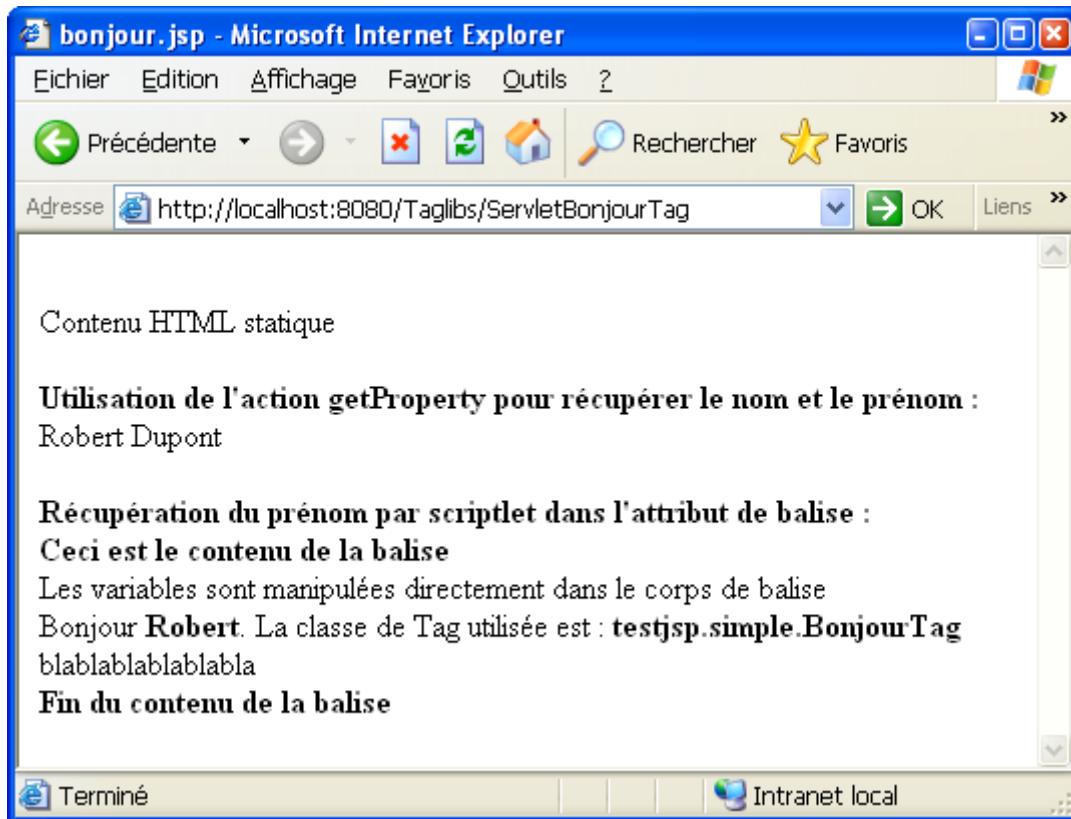
3) la balise a l'attribut liste en balise jsp gestion interne du corps variable script 0

gestion interne du corps variable script 1 gestion interne du corps variable script 2

nb éléments = 3

5.5. BALISES PERSONNALISEES (VERSION 1.2)

5.5.1. CORPS DE BALISE AVEC VARIABLES DE SCRIPT



5.5.1.1. La classe BonjourTag

```
package testjsp.simple;

import java.io.IOException;
import java.util.Date;
import java.text.DateFormat;
import javax.servlet.jsp.JspException;
import javax.servlet.jsp.tagext.TagSupport;

public class BonjourTag extends TagSupport {
    //Initialisation de nom obligatoire car l'attribut est facultatif
    private String nom="";
    private String prenom;

    //getters et setters pour les attributs d'instance
    public String getPrenom() {
        return prenom;
    }
    public void setPrenom(String prenom) {
        this.prenom = prenom;
    }

    public int doStartTag() throws JspException {
        pageContext.setAttribute("prenom", prenom);
        pageContext.setAttribute("classeTag", this.getClass().getName());
        return EVAL_BODY_INCLUDE;
    }
}
```

5.5.1.2. La classe BonjourTagExtraInfo

```

package testjsp.simple;

import javax.servlet.jsp.tagext.TagData;
import javax.servlet.jsp.tagext.TagExtraInfo;
import javax.servlet.jsp.tagext.VariableInfo;

public class BonjourTagExtraInfo extends TagExtraInfo {

    public VariableInfo[] getVariableInfo(TagData arg0) {
        return new VariableInfo[] {
            new VariableInfo
                ("prenom", "java.lang.String", true, VariableInfo.NESTED),
            new VariableInfo
                ("classeTag", "java.lang.String", true, VariableInfo.NESTED)
        };
    }
}

```

5.5.1.3. Le fichier exemples.tld

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE taglib PUBLIC "-//Sun Microsystems,
    Inc.//DTD JSP Tag Library 1.1//EN"
    "http://java.sun.com/j2ee/dtds/web-jsptaglibrary_1_1.dtd">

<taglib>
    <tlibversion>1.0</tlibversion>
    <jspversion>1.1</jspversion>
    <shortname>ex</shortname>
    <info>Exemples de bibliothèque de balises</info>
    <tag>
        <name>Bonjour</name>
        <tagclass>testjsp.simple.BonjourTag</tagclass>
        <teiclass>testjsp.simple.BonjourTagExtraInfo</teiclass>
        <bodycontent>JSP</bodycontent>
        <info>Exemple de balise simple</info>
        <attribute>
            <name>prenom</name>
            <required>true</required>
            <rtextprvalue>true</rtextprvalue>
        </attribute>
    </tag>
</taglib>

```

5.5.1.4. Le fichier bonjour.jsp

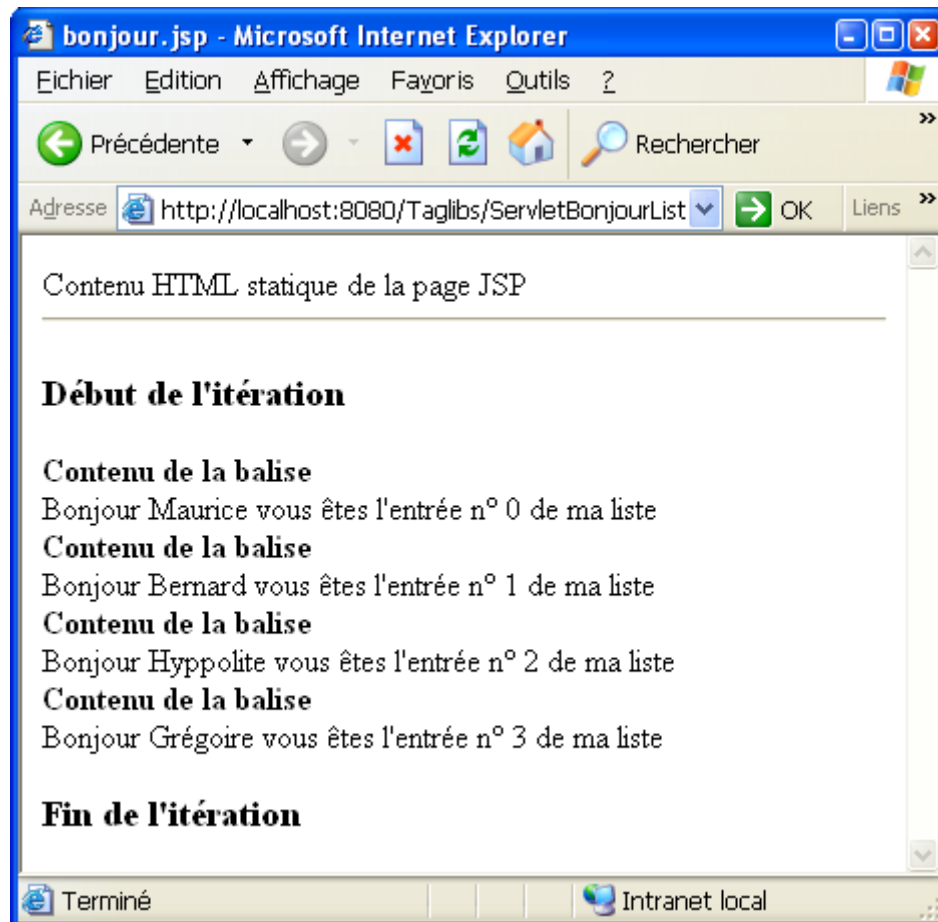
```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<%@taglib uri="/exemples" prefix="ex"%>
<jsp:useBean id="adherent" class="testjsp.simple.Adherent" scope="request"/>

<HTML>
  <HEAD>
    <TITLE>bonjour.jsp</TITLE>
  </HEAD>
  <BODY>
    <P><BR>Contenu HTML statique<BR></P>
    <P><B>Utilisation de l'action getProperty pour récupérer le
      nom et le prénom :</B><BR>
      <jsp:getProperty name="adherent" property="prenom"/>
      <jsp:getProperty name="adherent" property="nom"/>
    </P>
    <P><B> Récupération du prénom par scriptlet dans l'attribut
      de balise</B>
      <ex:Bonjour prenom="<%=adherent.getPrenom()%%">
        <BR><B>Ceci est le contenu de la balise</B>
        <BR>Les variables sont manipulées directement
        dans le corps de balise<BR>
        Bonjour <B><%=prenom%></B>.
        La classe de Tag utilisée est : <B><%=classeTag%></B>
        <BR>blablablablablabla<BR>
      </ex:Bonjour>
    </P>
  </BODY>
</HTML>

```

5.5.2. BALISE IMPLEMENTANT UNE ITERATION (VERSION 1.2)



5.5.2.1. Servlet mettant un objet liste à disposition

```
package testjsp.simple;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.util.ArrayList;

public class ServletBonjourListeTag extends HttpServlet {

    protected void service(HttpServletRequest req,
                           HttpServletResponse rep)
        throws ServletException, IOException {

        RequestDispatcher rd;
        ServletContext sc = getServletConfig().getServletContext();
        /**
         * Instanciation et remplissage de la liste,
         * puis définition de l'objet comme attribut de la
         * requête HTTP
         */
        ArrayList listeNoms = new ArrayList();
        listeNoms.add("Maurice");
        listeNoms.add("Bernard");
        listeNoms.add("Hyppolite");
        listeNoms.add("Grégoire");
        req.setAttribute("liste", listeNoms);

        rd=sc.getRequestDispatcher("/bonjourListe.jsp");
        rd.forward(req,rep);
    }
}
```

5.5.2.2. La classe BonjourListeTag

```

package testjsp.simple;

import java.io.IOException;
import java.util.ArrayList;
import javax.servlet.jsp.JspException;
import javax.servlet.jsp.JspException;
import javax.servlet.jsp.tagext.BodyContent;
import javax.servlet.jsp.tagext.BodyTagSupport;

public class BonjourListeTag extends BodyTagSupport {
    private ArrayList listeNoms;
    private int index;
    private StringBuffer sortie;
    /**
     * Constructeur pour BonjourListeTag.
     */
    public BonjourListeTag() {
        super();
        sortie = new StringBuffer();
    }
    /**
     * Getter et setter pour l'objet Liste
     */
    public ArrayList getListeNoms() {
        return listeNoms;
    }
    public void setListeNoms(ArrayList listeNoms) {
        this.listeNoms = listeNoms;
    }
    /** Cette méthode valorise les variables grâce à pageContext.
     * Elle est appelée une première fois en début de traitement de la
     * balise (doStartTag()) puis après chaque itération sur le corps
     * de la balise (doAfterBody()).
     */
    private void setNomSuivant() {
        pageContext.setAttribute("nom",
                                listeNoms.get(index).toString());
        pageContext.setAttribute("index", new Integer(index));
    }

    public int doStartTag() throws JspException {
        if (listeNoms.size() > 0) {
            setNomSuivant();
            return EVAL_BODY_AGAIN;
        } else {
            return SKIP_BODY;
        }
    }
}

```

.../...

```
/** Le moteur JSP appelle cette méthode chaque fois que
 * le contenu du corps de la balise aura été traité.
 * Si elle retourne SKIP_BODY, le contenu du corps aura été
 * traité pour la dernière fois. Si elle retourne
 * EVAL_BODY_TAG, le corps sera traité une nouvelle fois.
 * et cette méthode sera encore appelée une fois.
 */
public int doAfterBody() throws JspException {
    BodyContent monBody = getBodyContent();
    if (monBody != null) {
        sortie.append(monBody.getString());
        try {
            monBody.clear();
        } catch (IOException e) {
            throw new JspException("Erreur fatale d'E/S");
        }
    }
    if (++index < listeNoms.size()) {
        setNomSuivant();
        return EVAL_BODY_AGAIN;
    } else {
        return SKIP_BODY;
    }
}
/** Appelée une fois que le corps de la balise est traité.
 * Cette méthode permet de générer le flux de sortie avec le
 * contenu créé durant le traitement du corps.
 */
public int doEndTag() throws JspException {
    try {
        BodyContent monBody = getBodyContent();
        monBody.getEnclosingWriter().write(sortie.toString());
    } catch (IOException e) {
        throw new JspException("Erreur fatale d'E/S");
    }
    return EVAL_PAGE;
}
}
```


5.5.2.3. La classe BonjourListeTagExtralInfo

```

package testjsp.simple;

import javax.servlet.jsp.tagext.TagData;
import javax.servlet.jsp.tagext.TagExtralInfo;
import javax.servlet.jsp.tagext.VariableInfo;

public class BonjourListeTagExtralInfo extends TagExtralInfo {

    public VariableInfo[] getVariableInfo(TagData arg0) {
        return new VariableInfo[]{
            new VariableInfo(
                "nom", "java.lang.String", true, VariableInfo.NESTED),
            new VariableInfo(
                "index", "java.lang.Integer", true, VariableInfo.NESTED)
        };
    }
}

```

5.5.2.4. Le fichier bonjourListe.jsp

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<%@taglib uri="/exemples" prefix="ex"%>
<%@page import="java.util.*"%>
<jsp:useBean id="liste" class="java.util.List" scope="request"/>

<HTML>
  <HEAD>
    <TITLE>bonjour.jsp</TITLE>
  </HEAD>
  <BODY>

    Contenu HTML statique de la page JSP
    <HR>
    <H3>Début de l'itération</H3>

    <ex:BonjourListe listeNoms="<%=liste%>">
      <B>Contenu de la balise</B><BR>
      Bonjour <%=nom%> vous êtes l'entrée n° <%=index%> de ma liste
    </ex:BonjourListe>

    <H3>Fin de l'itération</H3>
  </BODY>
</HTML>

```

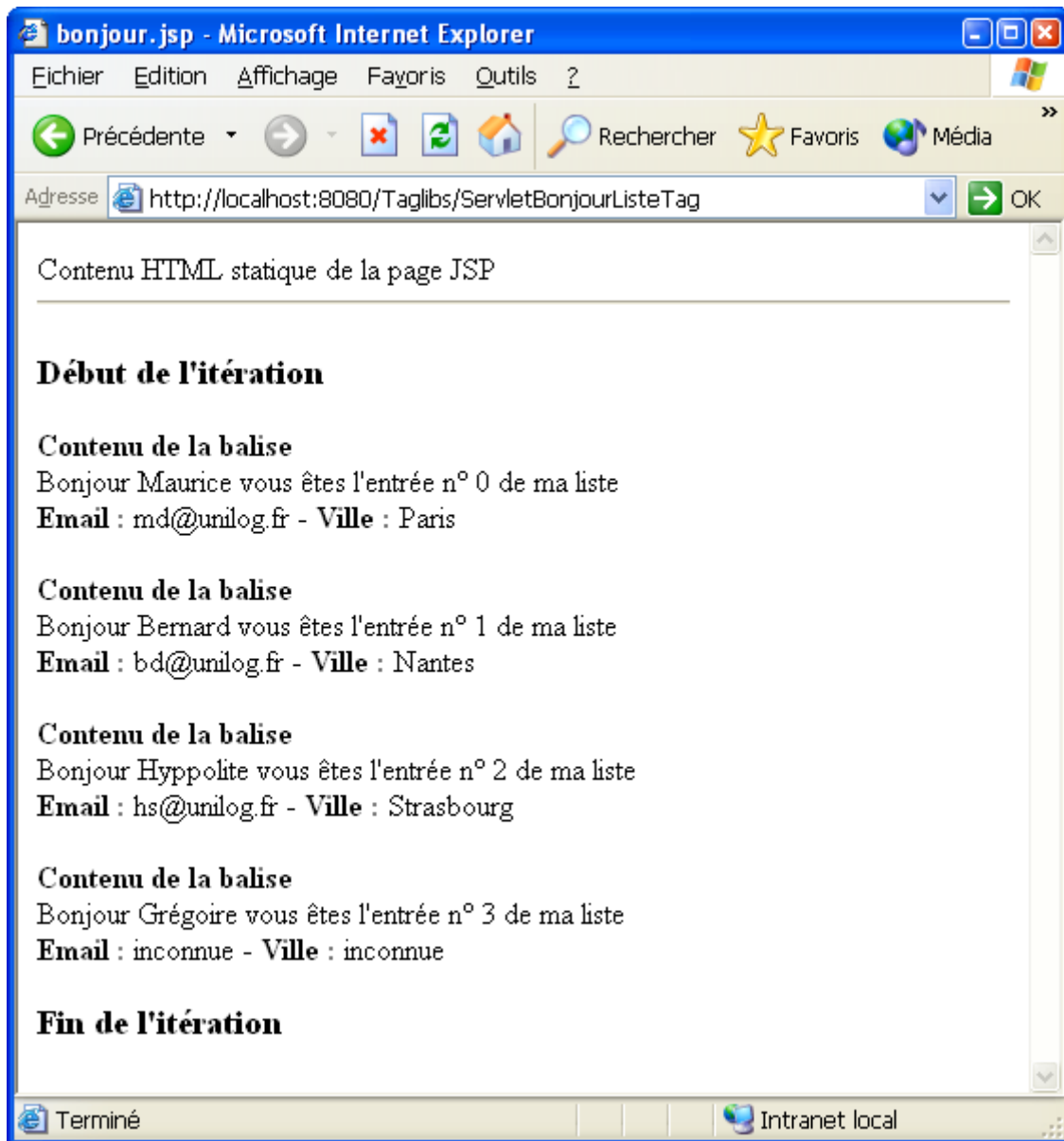
5.5.2.5. Le fichier exemples.tld

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE taglib PUBLIC "-//Sun Microsystems, Inc.//DTD JSP Tag Library 1.1//EN"
"http://java.sun.com/j2ee/dtds/web-jsptaglibrary_1_1.dtd">

<taglib>
  <tlibversion>1.0</tlibversion>
  <jspversion>1.1</jspversion>
  <shortname>ex</shortname>
  <info>Exemples de bibliothèque de balises</info>
  <tag>
    <name>Bonjour</name>
    <tagclass>testjsp.simple.BonjourTag</tagclass>
    <teiclass>testjsp.simple.BonjourTagExtraInfo</teiclass>
    <bodycontent>JSP</bodycontent>
    <info>Exemple de balise simple</info>
    <attribute>
      <name>prenom</name>
      <required>true</required>
      <rteprvalue>true</rteprvalue>
    </attribute>
  </tag>
  <tag>
    <name>BonjourListe</name>
    <tagclass>testjsp.simple.BonjourListeTag</tagclass>
    <teiclass>testjsp.simple.BonjourListeTagExtraInfo</teiclass>
    <bodycontent>JSP</bodycontent>
    <info>Exemple de balise avec itération</info>
    <attribute>
      <name>listeNoms</name>
      <required>true</required>
      <rteprvalue>true</rteprvalue>
    </attribute>
  </tag>
</taglib>
```

5.5.3. IMBRICATION DE BALISES (VERSION 1.2)



5.5.3.1. Servlet qui crée la liste

```

package testjsp.simple;

import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class ServletBonjourListeTag extends HttpServlet {

    protected void service(HttpServletRequest req, HttpServletResponse rep)
        throws ServletException, IOException {
        RequestDispatcher rd;
        ServletContext sc = getServletContext().getServletContext();
        /**
         * Instanciation d'un objet LinkedList, remplissage de
         * la liste, puis définition de l'objet liste comme
         * attribut de la requête http.
         */
        List listeNoms = new LinkedList();
        listeNoms.add("Maurice");
        listeNoms.add("Bernard");
        listeNoms.add("Hyppolite");
        listeNoms.add("Grégoire");
        req.setAttribute("listeNoms",listeNoms);
        /**
         * Même manipulation avec une hashtable contenant des Adherents
         */
        Hashtable listeAdhs = new Hashtable();
        listeAdhs.put("Maurice", new Adherent("DUPONT", "Maurice",
                                                "Paris", "md@unilog.fr"));
        listeAdhs.put("Bernard", new Adherent("DURAND", "Bernard",
                                                "Nantes", "bd@unilog.fr"));
        listeAdhs.put("Hyppolite", new Adherent("MARTIN", "Hyppolite",
                                                "Lyon", "hs@unilog.fr"));
        listeAdhs.put("Robert", new Adherent("MOREAU", "Robert",
                                                "Toulouse", "rm@unilog.fr"));
        req.setAttribute("listeAdhs",listeAdhs);

        rd=sc.getRequestDispatcher("/bonjourListe.jsp");
        rd.forward(req,rep);
    }
}

```

5.5.3.2. Modification de la classe BonjourListeTag

```
package testjsp.simple;

import java.io.IOException;
import java.util.*;
import javax.servlet.jsp.JspException;
import javax.servlet.jsp.JspException;
import javax.servlet.jsp.tagext.BodyContent;
import javax.servlet.jsp.tagext.BodyTagSupport;

public class BonjourListeTag extends BodyTagSupport {

    [...]

    /** Méthode supplémentaire implémentée pour mettre le nom courant
     * à disposition de la balise fille.
     */
    public String getNomCourant() {
        String nom = (String) listeNoms.get(index);
        return nom;
    }

    [...]

}
```

5.5.3.3. La classe InfosAdherentTag

```

package testjsp.simple;

import java.util.Hashtable;
import javax.servlet.jsp.JspException;
import javax.servlet.jsp.JspException;
import javax.servlet.jsp.tagext.TagSupport;

public class InfosAdherentTag extends TagSupport {
    private Hashtable listeAdherents;
    /**getter et setter pour la Hashtable listeAdherents */
    public Hashtable getListeAdherents() {
        return listeAdherents;
    }
    public void setListeAdherents(Hashtable listeAdhs) {
        this.listeAdherents = listeAdhs;
    }
    /** La méthode vérifie si la balise possède un ancêtre du type requis qui pourra être
    utilisé pour récupérer le prénom à rechercher.
    * La méthode statique findAncestorWithClass() est plus souple que getParent().
    * Elle permet de récupérer la balise du type recherché même s'il ne s'agit pas du
    parent direct de cette balise (Il peut y avoir
    * imbrication à plusieurs niveaux...)
    */
    public int doStartTag() throws JspException {
        //Initialisation des variables email et ville
        String email = "inconnue";
        String ville = "inconnue";

        BonjourListeTag blt = (BonjourListeTag)
            TagSupport.findAncestorWithClass(this, BonjourListeTag.class);

        if (blt == null) {
            throw new JspException("DetailAdherentTag ne peut être
            utilisé qu'imbriqué dans une balise BonjourListeTag");
        }

        Adherent adh= (Adherent)listeAdherents.get(blt.getNomCourant());
        if (adh != null){
            email = adh.getEmail();
            ville = adh.getVille();
        }
        pageContext.setAttribute("email", email);
        pageContext.setAttribute("ville", ville);
        return EVAL_BODY_INCLUDE;
    }
}

```

5.5.3.4. La classe InfosAdherentTagExtraInfo

```
package testjsp.simple;

import javax.servlet.jsp.tagext.TagData;
import javax.servlet.jsp.tagext.TagExtraInfo;
import javax.servlet.jsp.tagext.VariableInfo;

public class InfosAdherentTagExtraInfo extends TagExtraInfo {

    public VariableInfo[] getVariableInfo(TagData arg0) {
        return new VariableInfo[]{
            new VariableInfo(
                "email", "java.lang.String", true, VariableInfo.NESTED),
            new VariableInfo(
                "ville", "java.lang.String", true, VariableInfo.NESTED)
        };
    }
}
```

5.5.3.5. Le fichier bonjourListe.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<%@taglib uri="/exemples" prefix="ex"%>
<%@page import="java.util.*"%>
<jsp:useBean id="listeNoms" class="java.util.List" scope="request"/>
<jsp:useBean id="listeAdhs" class="java.util.Hashtable" scope="request"/>
<HTML>
  <HEAD>
    <TITLE>bonjour.jsp</TITLE>
  </HEAD>
  <BODY>
    Contenu HTML statique de la page JSP
    <HR>
    <H3>Début de l'itération</H3>
    <ex:BonjourListe listeNoms="<%=listeNoms%>">
      <B>Contenu de la balise</B><BR>
      Bonjour <%=nom%> vous êtes l'entrée n° <%=index%> de ma liste<BR>
      <ex:DetailAdherent listeAdherents="<%=listeAdhs %%">
        <B>Email : </B> <%=email%> - <B>Ville : </B><%=ville%><BR>
      </ex:DetailAdherent>
      <BR>
    </ex:BonjourListe>
    <H3>Fin de l'itération</H3>
  </BODY>
</HTML>
```

5.5.3.6. Le fichier exemples.tld

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE taglib PUBLIC "-//Sun Microsystems, Inc.//DTD JSP Tag Library 1.1//EN"
    "http://java.sun.com/j2ee/dtds/web-jsptaglibrary_1_1.dtd">

<taglib>
  <tlibversion>1.0</tlibversion>
  <jspversion>1.1</jspversion>
  <shortname>ex</shortname>
  <info>Exemples de bibliothèque de balises</info>

  [...]

  <tag>
    <name>BonjourListe</name>
    <tagclass>testjsp.simple.BonjourListeTag</tagclass>
    <teiclass>testjsp.simple.BonjourListeTagExtraInfo</teiclass>
    <bodycontent>JSP</bodycontent>
    <info>Exemple de balise avec itération</info>
    <attribute>
      <name>listeNoms</name>
      <required>true</required>
      <rteprvalue>true</rteprvalue>
    </attribute>
  </tag>
  <tag>
    <name>InfosAdherent</name>
    <tagclass>testjsp.simple.InfosAdherentTag</tagclass>
    <teiclass>testjsp.simple.InfosAdherentTagExtraInfo</teiclass>
    <bodycontent>JSP</bodycontent>
    <info>Exemple de balise imbriquée</info>
    <attribute>
      <name>listeAdherents</name>
      <required>true</required>
      <rteprvalue>true</rteprvalue>
    </attribute>
  </tag>
</taglib>
```


6. UTILISATION DE BALISES JSTL

6.1. SAISIE CONDITIONNELLE

Nous voulons qu'une seule et même JSP affiche soit :

Bonjour Monsieur Untel

soit :

Bonjour Madame Untel

quel est votre nom de jeune fille :

en fonction de la civilité de la personne.

6.1.1. LA CLASSE « CLIENT »

```
package pack;

public class Client implements java.io.Serializable
{
    private String civilite;
    private String nom;
    private String nomJF;

    public Client(String pCivil, String pNom)
    {
        civilite = pCivil;
        nom = pNom;
    }

    public String getCivilite() {    return civilite;    }

    public void setCivilite(String string) {    civilite = string;    }

    public String getNom() {    return nom;    }

    public void setNom(String string)    {    nom = string;    }

    public String getNomJF() {    return nomJF;    }

    public void setNomJF(String string) {    nomJF = string;    }

    public String toString() { return civilite + " " + nom;    }
}
```

Notez la surcharge de la méthode « toString() » pour faciliter l'expression dans la JSP.

6.1.2. LE SERVLET

```
package pack;

import java.io.IOException;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class CtrlEssais extends HttpServlet
{
    protected void service(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException
    {
        RequestDispatcher rd = null;

        req.setAttribute("client", new Client("Madame","Untel"));
        rd = getServletContext().getRequestDispatcher("/essai1");

        rd.forward(req,resp);
    }
}
```

Pour tester les différents cas, il suffit de remplacer « Madame » par « Mademoiselle » puis « Monsieur ».

6.1.3. LA JSP

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<HTML>
<HEAD>

  <%@ taglib uri="http://java.sun.com/jstl/core" prefix="c" %>

  <%@ page
    language="java"
    contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"
  %>
  <META http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
  <META name="GENERATOR" content="IBM WebSphere Studio">
  <TITLE>essai1.jsp</TITLE>
</HEAD>
<BODY>
  <form method="get"
    action="http://localhost:9080/jstlWAR/CtrlEssais">
    <br/><br/><br/>
    <center>

      <h2>Bonjour <c:out value="${requestScope.client}" /></h2>

      <c:if test="${requestScope.client.civilite == 'Madame'}">
        <br/><br/><br/>
        quel est votre nom de jeune fille :
        <input type="text" name="nomJF"/>
        <br/><br/><br/>
        <input type="submit"/>
      </c:if>
    </center>
  </form>
</BODY>
</HTML>
```

6.1.4. LE FICHIER WEB.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
"http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app id="WebApp">
  <display-name>jstlWAR</display-name>
  <servlet>
    <servlet-name>essai1</servlet-name>
    <display-name>essai1</display-name>
    <jsp-file>/essai1.jsp</jsp-file>
  </servlet>
  <servlet>
    <servlet-name>CtrlEssais</servlet-name>
    <display-name>CtrlEssais</display-name>
    <servlet-class>pack.CtrlEssais</servlet-class>
  </servlet>

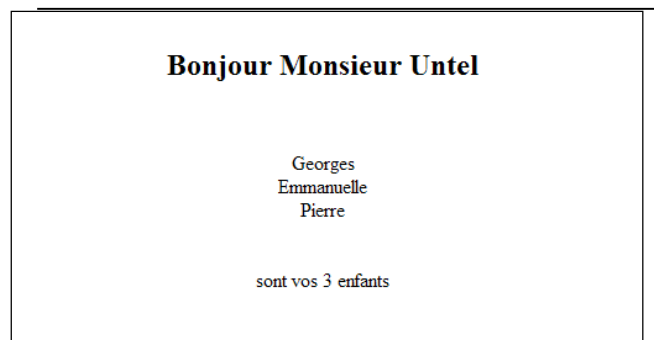
  <servlet-mapping>
    <servlet-name>essai1</servlet-name>
    <url-pattern>/essai1</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>CtrlEssais</servlet-name>
    <url-pattern>/CtrlEssais</url-pattern>
  </servlet-mapping>

  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>

  <taglib>
    <taglib-uri>http://java.sun.com/jstl/core</taglib-uri>
    <taglib-location>/WEB-INF/lib/standard.jar</taglib-location>
  </taglib>
</web-app>
```

6.2. BOUCLE SUR UNE LISTE

Nous voulons obtenir :



6.2.1. AJOUTS DANS LA CLASSE « CLIENT »

```
package pack ;

import java.util.ArrayList;

public class Client implements java.io.Serializable
{
    ...
    private ArrayList listeEnfants;

    public Client(String pCivil, String pNom)
    {
        ...
        listeEnfants = new ArrayList();
    }

    public ArrayList getListeEnfants()
    {
        return listeEnfants;
    }
    public void setListeEnfants(ArrayList list)
    {
        listeEnfants = list;
    }

    public void enregistreEnfant(String prenom)
    {
        listeEnfants.add(prenom);
    }
    ...
}
```

6.2.2. AJOUTS DANS LE SERVLET

```
...
RequestDispatcher rd = null;

Client client = new Client("Monsieur", "Untel");
client.enregistreEnfant("Georges");
client.enregistreEnfant("Emmanuelle");
client.enregistreEnfant("Pierre");
req.setAttribute("client", client);

rd = getServletContext().getRequestDispatcher("/essai1");
...
```

6.2.3. AJOUTS DANS LA JSP

```
<c:forEach    var="enfant"
              items="${requestScope.client.listeEnfants}"
              varStatus="vs">

    <c:out value="${enfant}"/><br/>
    <c:if test="${vs.last}">
        <br/><br/>
        <c:out value="sont vos ${vs.count} enfants"/>
    </c:if>

</c:forEach>
```

Pas de changement dans le web.xml.

6.3. IMPORT

Vous devez importer une JSP dans une autre, en lui passant un paramètre. Vous prendrez en compte le fait qu'une exception peut être levée.

6.3.1. LA PAGE QUI IMPORTE L'AUTRE

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<HTML>
<HEAD>
<%@ taglib uri="http://java.sun.com/jstl/core" prefix="c" %>
<%@ page
language="java"
contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"
%>
<META http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<META name="GENERATOR" content="IBM WebSphere Studio">
<TITLE>essaiImport.jsp</TITLE>
</HEAD>
<BODY>
Première page <br/>

<c:catch var="pepin">
  <c:import url="/pageIncluse">
    <c:param name="titre" value="Le titre"/>
  </c:import>
</c:catch>

<c:if test="${! empty pepin}">
  <c:out value="pépin : ${pepin}"/><br/>
</c:if>

<br/>Suite de la première page
</BODY>
</HTML>
```

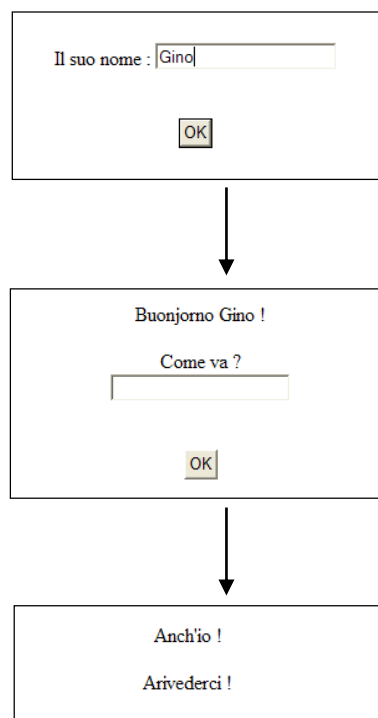
6.3.2. LA PAGE IMPORTEE

```
<%@ taglib uri="http://java.sun.com/jstl/core" prefix="c" %>
<br/>Page incluse : <br/>
paramètre reçu : <c:out value="${param['titre']}"/>
<br/>
```


6.4. INTERNATIONALISATION

6.4.1. CHOIX DE LA LANGUE INITIALE

Nous voulons définir la langue des différents écrans de notre site. Par exemple, l'italien :



6.4.1.1. Les fichiers .properties

Le fichier par défaut : « msg.properties »

```
# QUESTIONS posées
question.qui=Votre nom :
question.langue=Choisissez votre langue :
question.commentVa=Comment allez-vous ?

# MESSAGES affichés
msg.bonjour=Bonjour {0} !
msg.reponse=Moi aussi !
msg.auRevoir=A bientôt !

# LIBELLES
lbl.soumettre=Envoi
```

Le fichier en italien : « msg_it.properties »

```
# QUESTIONS posées
question.qui=Il suo nome :
question.langue=Sceglie la sua lingua :
question.commentVa=Come va ?

# MESSAGES affichés
msg.bonjour=Buongiorno {0} !
msg.reponse=Anch'io !
msg.auRevoir=Arivederci !

# LIBELLES
lbl.soumettre=OK
```

Le fichier en anglais : « msg_en.properties »

```
# QUESTIONS posées
question.qui=Your name :
question.langue=Choose your language :
question.commentVa=How are you ?

# MESSAGES affichés
msg.bonjour=Hello {0} !
msg.reponse=Me too !
msg.auRevoir=Bye !

# LIBELLES
lbl.soumettre=OK
```

6.4.1.2. Le fichier web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application
2.3//EN" "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app id="WebApp">
  <display-name>JSTLinternatWAR</display-name>

  <context-param>
    <param-name>
      javax.servlet.jsp.jstl.fmt.localizationContext
    </param-name>
    <param-value>langues/msg</param-value>
  </context-param>

  <context-param>
    <param-name>
      javax.servlet.jsp.jstl.fmt.fallbackLocale
    </param-name>
    <param-value>it</param-value>
  </context-param>

  <servlet>
    <servlet-name>login</servlet-name>
    <display-name>login</display-name>
    <jsp-file>/login.jsp</jsp-file>
  </servlet>
  <servlet>
    <servlet-name>poseQuestion</servlet-name>
    <display-name>poseQuestion</display-name>
    <jsp-file>/poseQuestion.jsp</jsp-file>
  </servlet>
  <servlet>
    <servlet-name>fin</servlet-name>
    <display-name>fin</display-name>
    <jsp-file>/fin.jsp</jsp-file>
  </servlet>
  <servlet-mapping>
    <servlet-name>login</servlet-name>
    <url-pattern>/login</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>poseQuestion</servlet-name>
    <url-pattern>/poseQuestion</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>fin</servlet-name>
    <url-pattern>/fin</url-pattern>
  </servlet-mapping>
  <welcome-file-list>
    <welcome-file>login.jsp</welcome-file>
  </welcome-file-list>
  <taglib>
    <taglib-uri>http://java.sun.com/jstl/fmt</taglib-uri>
    <taglib-location>/WEB-INF/lib/standard.jar</taglib-location>
  </taglib>
</web-app>
```

6.4.1.3. Les JSP

login.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<HTML>
<HEAD>

<%@ taglib uri="http://java.sun.com/jstl/fmt" prefix="fmt" %>

<%@ page language="java"
      contentType="text/html; charset=ISO-8859-1"
      pageEncoding="ISO-8859-1"
%>
<META http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<META name="GENERATOR" content="IBM WebSphere Studio">
<TITLE>login.jsp</TITLE>
</HEAD>
<BODY>
  <form action="http://localhost:9080/JSTLinternatWAR/poseQuestion">
    <br/><br/><br/>
    <center>

      <fmt:message key="question.qui" />

      <input type="text" name="nom" />
      <br/><br/><br/>

      <input type="submit"
            value="<fmt:message key='lbl.soumettre' />" />
    </center>
  </form>
</BODY>
</HTML>
```

poseQuestion.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<HTML>
<HEAD>
<%@ taglib uri="http://java.sun.com/jstl/fmt" prefix="fmt" %>
<%@ page language="java"
      contentType="text/html; charset=ISO-8859-1"
      pageEncoding="ISO-8859-1"
%>
<META http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<META name="GENERATOR" content="IBM WebSphere Studio">
<TITLE>poseQuestion.jsp</TITLE>
</HEAD>
<BODY>
<form method="get" action="http://localhost:9080/JSTLinternatWAR/fin">
  <center>

    <fmt:message key="msg.bonjour">
      <fmt:param value="{param['nom']}" />
    </fmt:message>

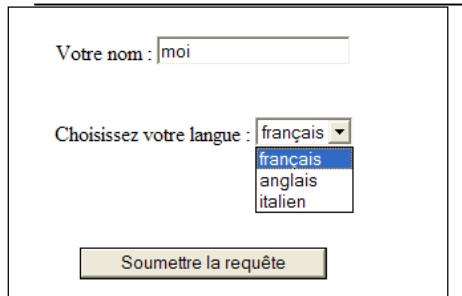
    <br/><br/>
    <fmt:message key="question.commentVa" />
    <br/>
    <input type="text" name="santé" />
    <br/><br/><br/>
    <input type="submit" value="<fmt:message key='lbl.soumettre' />" />
  </center>
</form>
</BODY>
</HTML>
```

fin.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<HTML>
<HEAD>
<%@ taglib uri="http://java.sun.com/jstl/fmt" prefix="fmt" %>
<%@ page language="java"
      contentType="text/html; charset=ISO-8859-1"
      pageEncoding="ISO-8859-1"
%>
<META http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<META name="GENERATOR" content="IBM WebSphere Studio">
<TITLE>fin.jsp</TITLE>
</HEAD>
<BODY>
  <center>
    <fmt:message key="msg.reponse" />
    <br/><br/>
    <fmt:message key="msg.auRevoir" />
  </center>
</BODY>
</HTML>
```

6.4.2. CHOIX DE LA LANGUE PAR L'UTILISATEUR

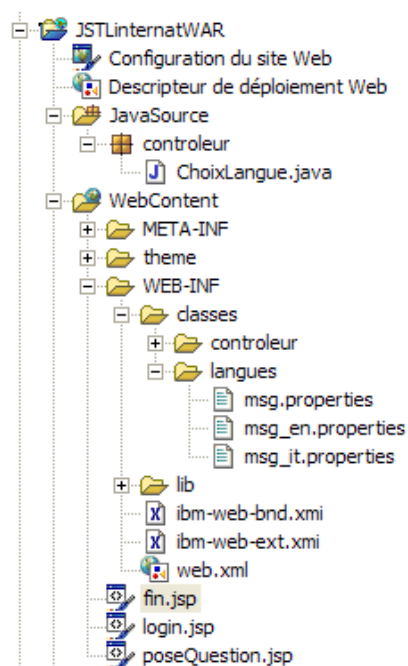
Nous voulons que, sur la page de connexion, l'utilisateur choisisse sa langue. Par exemple :



The screenshot shows a web form with two main sections. The first section is labeled 'Votre nom : ' followed by a text input field containing the text 'moi'. The second section is labeled 'Choisissez votre langue : ' followed by a dropdown menu. The dropdown menu is currently open, showing three options: 'français' (highlighted in blue), 'anglais', and 'italien'. Below these sections is a button labeled 'Soumettre la requête'.

Les pages suivantes sont fonction de son choix.

6.4.2.1. Structure de l'application



Pas de changement dans les fichiers .properties.

Dans le web.xml, simple ajout du servlet « choixLangue » et retrait de la définition d'une langue initiale.

6.4.2.2. La JSP de connexion

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<HTML>
<HEAD>

<%@ taglib uri="http://java.sun.com/jstl/fmt" prefix="fmt" %>

<%@ page language="java"
      contentType="text/html; charset=ISO-8859-1"
      pageEncoding="ISO-8859-1"
%>
<META http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<META name="GENERATOR" content="IBM WebSphere Studio">
<TITLE>login.jsp</TITLE>
</HEAD>
<BODY>
  <form action="http://localhost:9080/JSTLinternatWAR/ChoixLangue">
    <br/><br/><br/>
    <center>

      <fmt:message key="question.qui" />

      <input type="text" name="nom" />
      <br/><br/><br/>

      <fmt:message key="question.langue" />

      <select name="langue">
        <option value="_fr">français</option>
        <option value="_en">anglais</option>
        <option value="_it">italien</option>
      </select>
      <br/><br/><br/><br/><br/>
      <input type="submit"/>
    </center>
  </form>
</BODY>
</HTML>
```

6.4.2.3. Le servlet

```

package controleur;

import java.io.IOException;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

public class ChoixLangue extends HttpServlet
{
    protected void service(HttpServletRequest req,
                           HttpServletResponse resp)
        throws ServletException, IOException
    {
        /* Récupère le nom de base et l'emplacement des fichiers */
        /* .propriétés définis en paramètre de contexte dans le web.xml */
        String nomFic = getServletContext().getInitParameter(
            "javax.servlet.jsp.jstl.fmt.localizationContext");

        /* Récupère le suffixe correspondant à la langue choisie */
        String langueChoisie = req.getParameter("langue");
        nomFic += langueChoisie;
        /* S'il n'y en a pas, fichier sans suffixe (en français) */

        /* Définit le nom du fichier comme attribut de session */
        HttpSession session = req.getSession(true);
        session.setAttribute("ficMsg", nomFic);

        /* Chaîne avec la JSP suivante */
        RequestDispatcher rd =
            getServletContext().getRequestDispatcher("/poseQuestion");
        rd.forward(req,resp);
    }
}

```

6.4.2.4. Les 2 autres JSP

Les autres JSP « poseQuestion » et « fin » doivent, avant toute utilisation de la balise « message », ajouter :

```
<fmt:setBundle basename="${sessionScope.ficMsg}"/>
```